CS 540 Introduction to Artificial Intelligence

Perceptron

University of Wisconsin-Madison

Fall 2022

Part I: Single-layer Neural Network

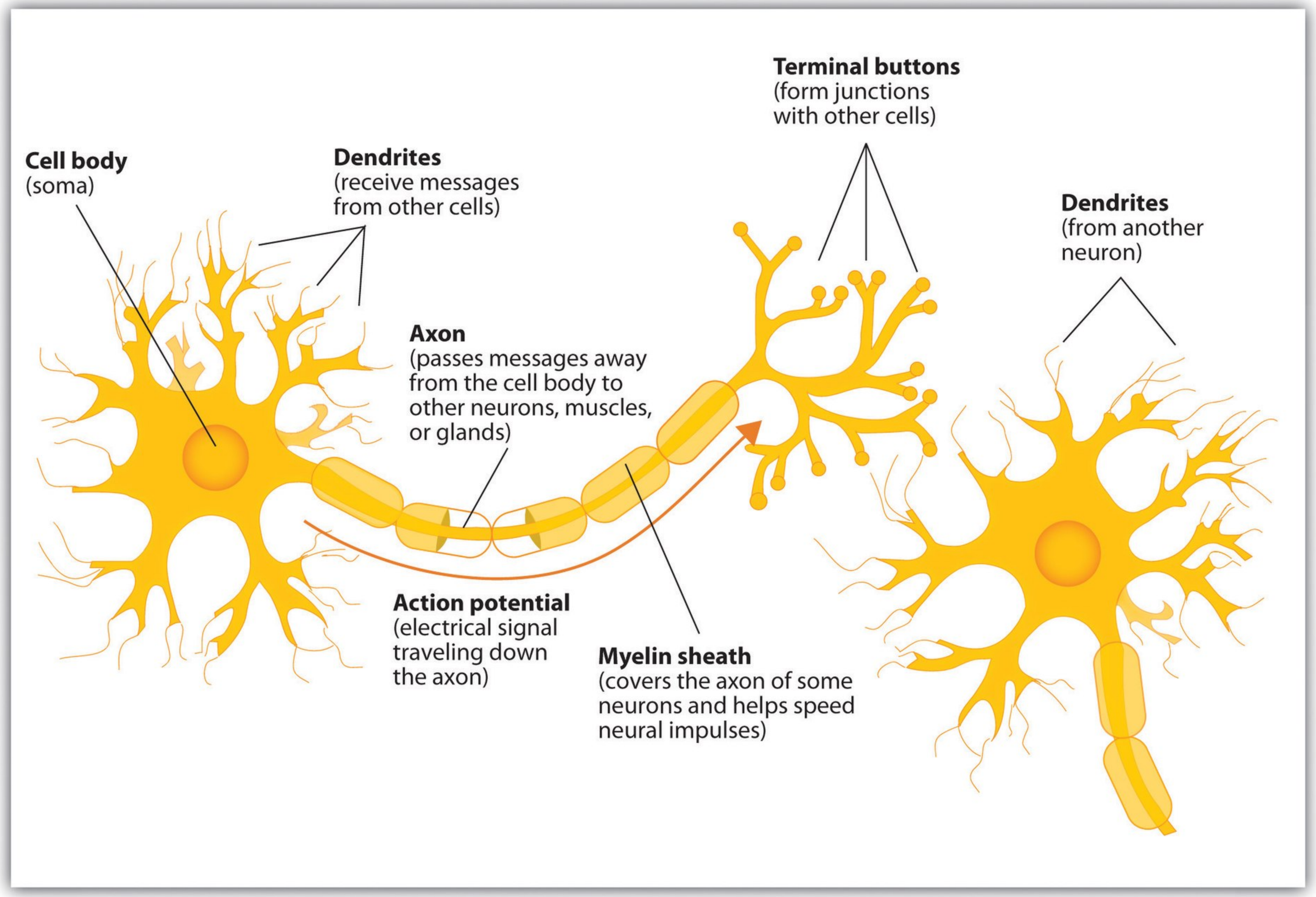
How to classify

Cats vs. dogs?

Inspiration from neuroscience

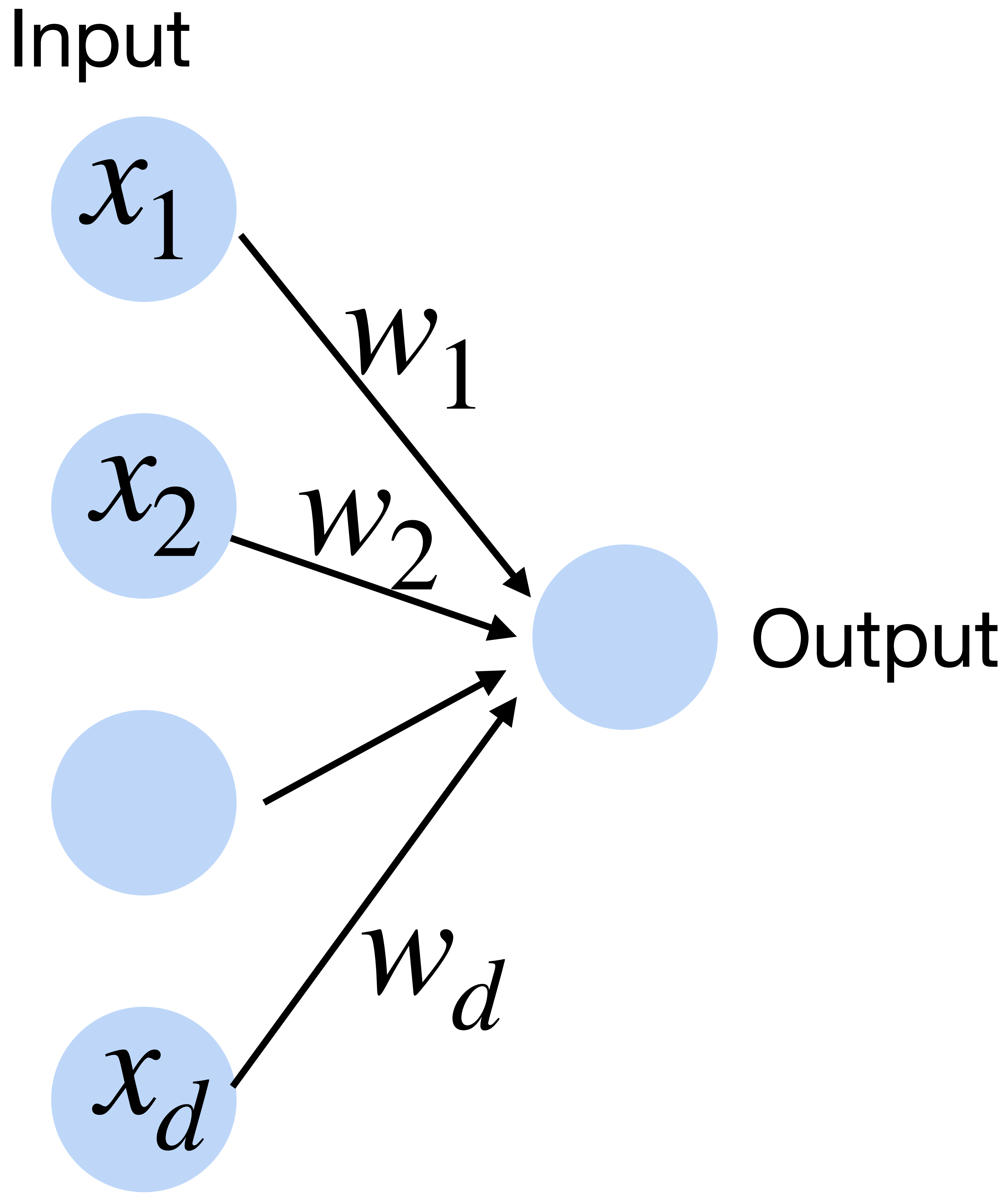
- Inspirations from human brains
- Networks of simple and homogenous units

(wikipedia)


Terminal buttons
(form junctions with other cells)
Cell body
(soma)
Dendrites
(receive messages from other cells)
Dendrites
(from another neuron)
Axon
(passes messages away from the cell body to other neurons, muscles, or glands)
Action potential
(electrical signal traveling down the axon)
Myelin sheath
(covers the axon of some neurons and helps speed neural impulses)

Perceptron

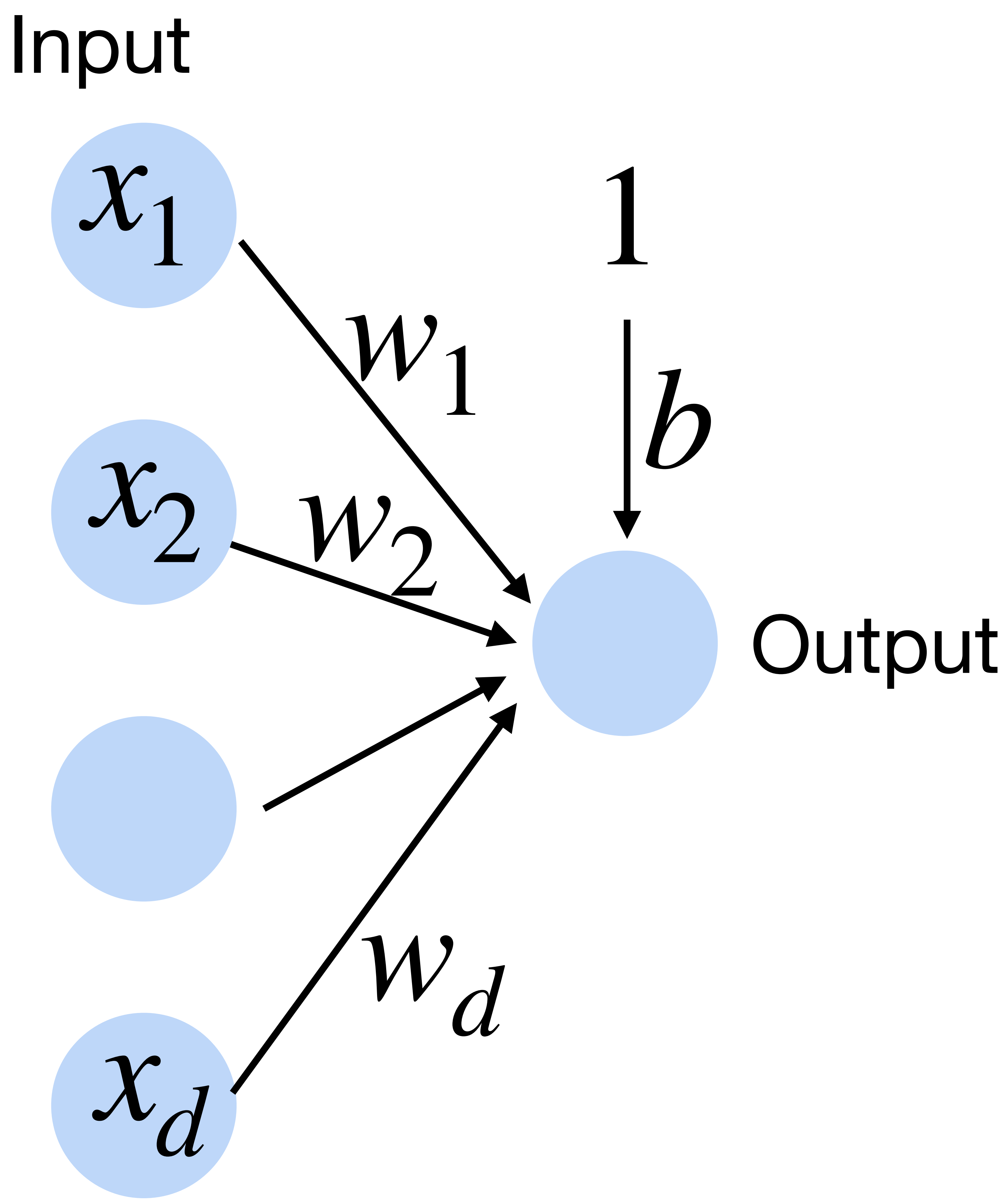
Cats vs. dogs?



Linear Perceptron

- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$f = \langle \mathbf{w}, \mathbf{x} \rangle + b$$



Cats vs. dogs?

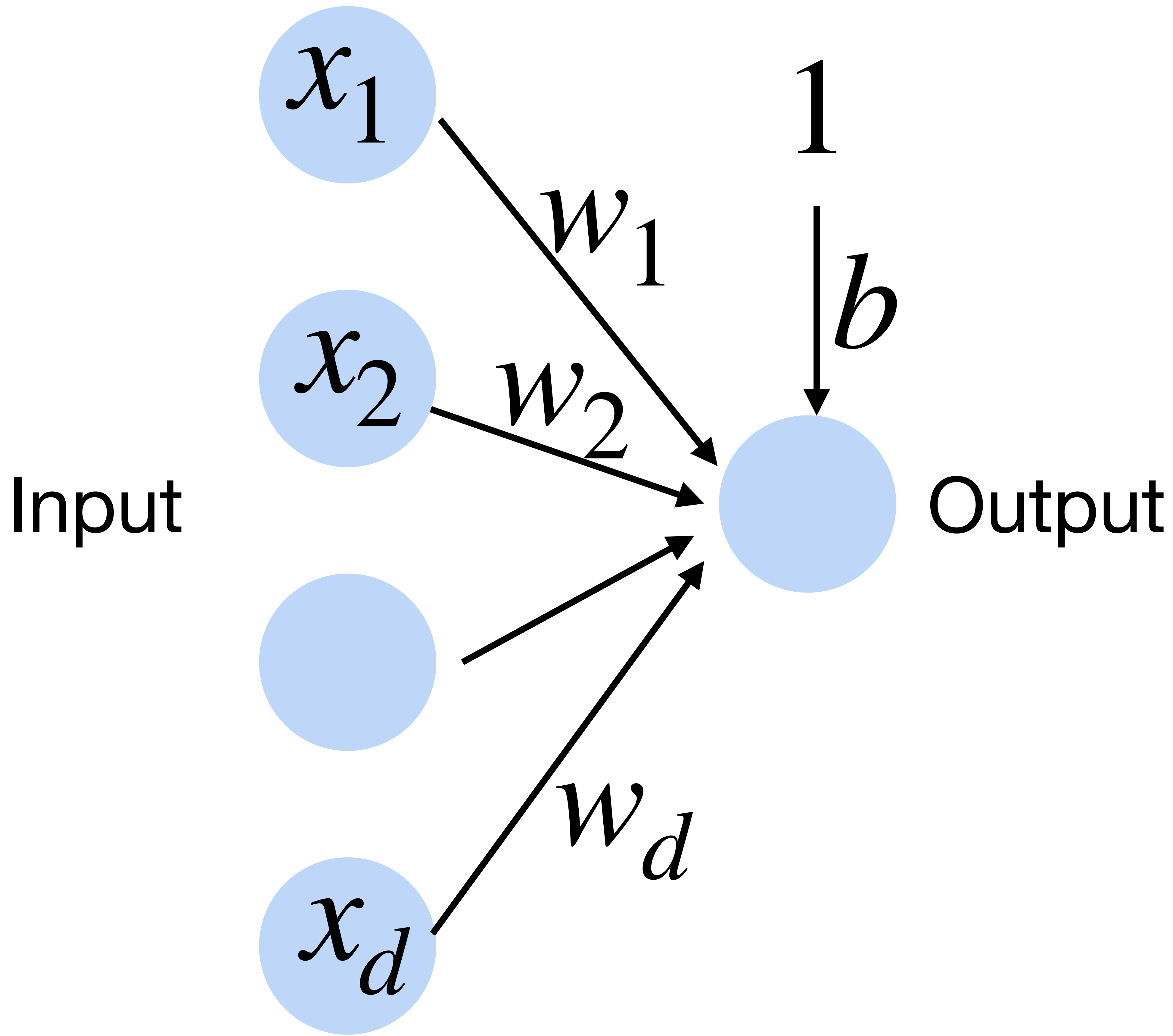
Perceptron

- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \qquad \sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Activation function

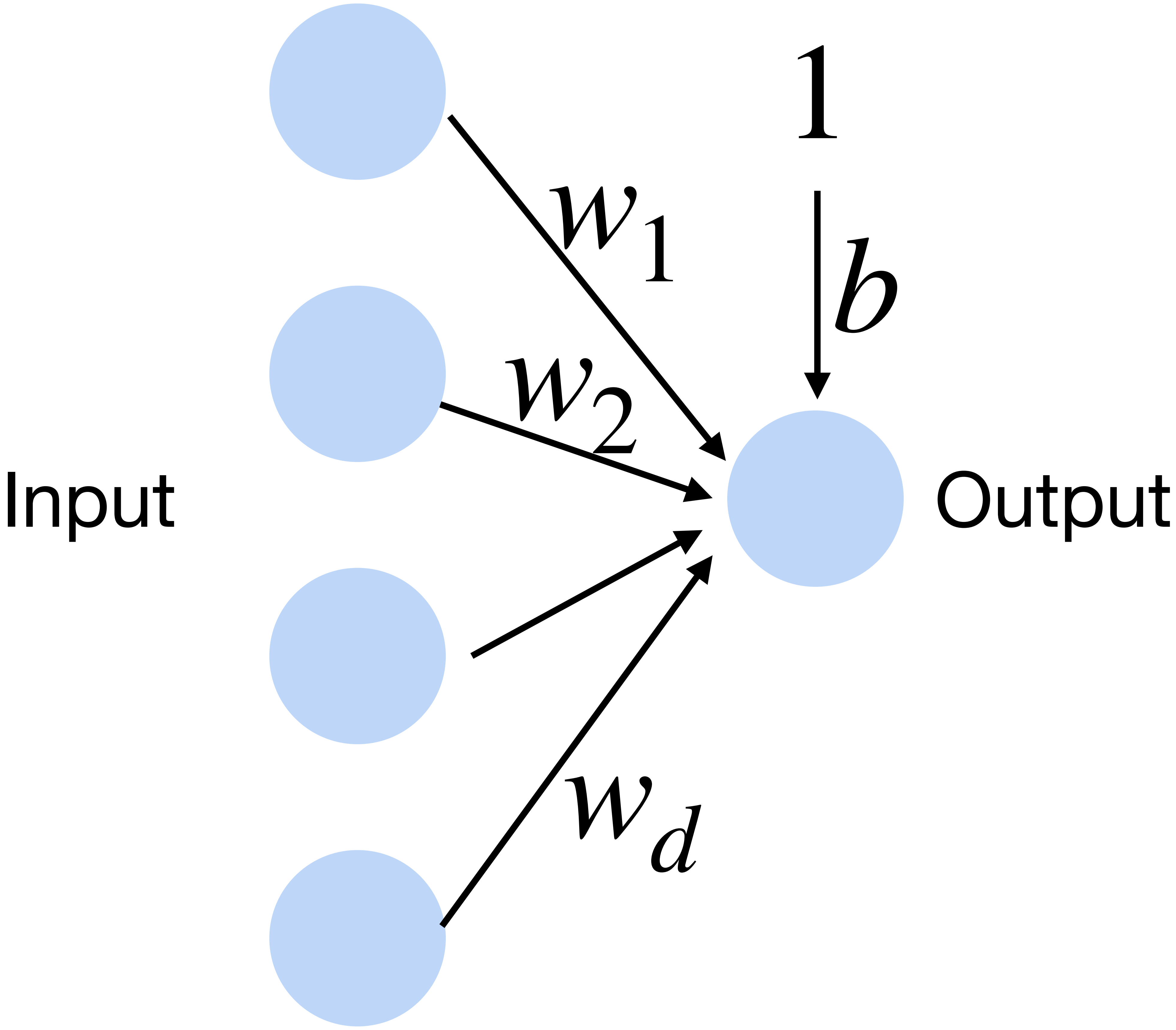
Cats vs. dogs?



Perceptron

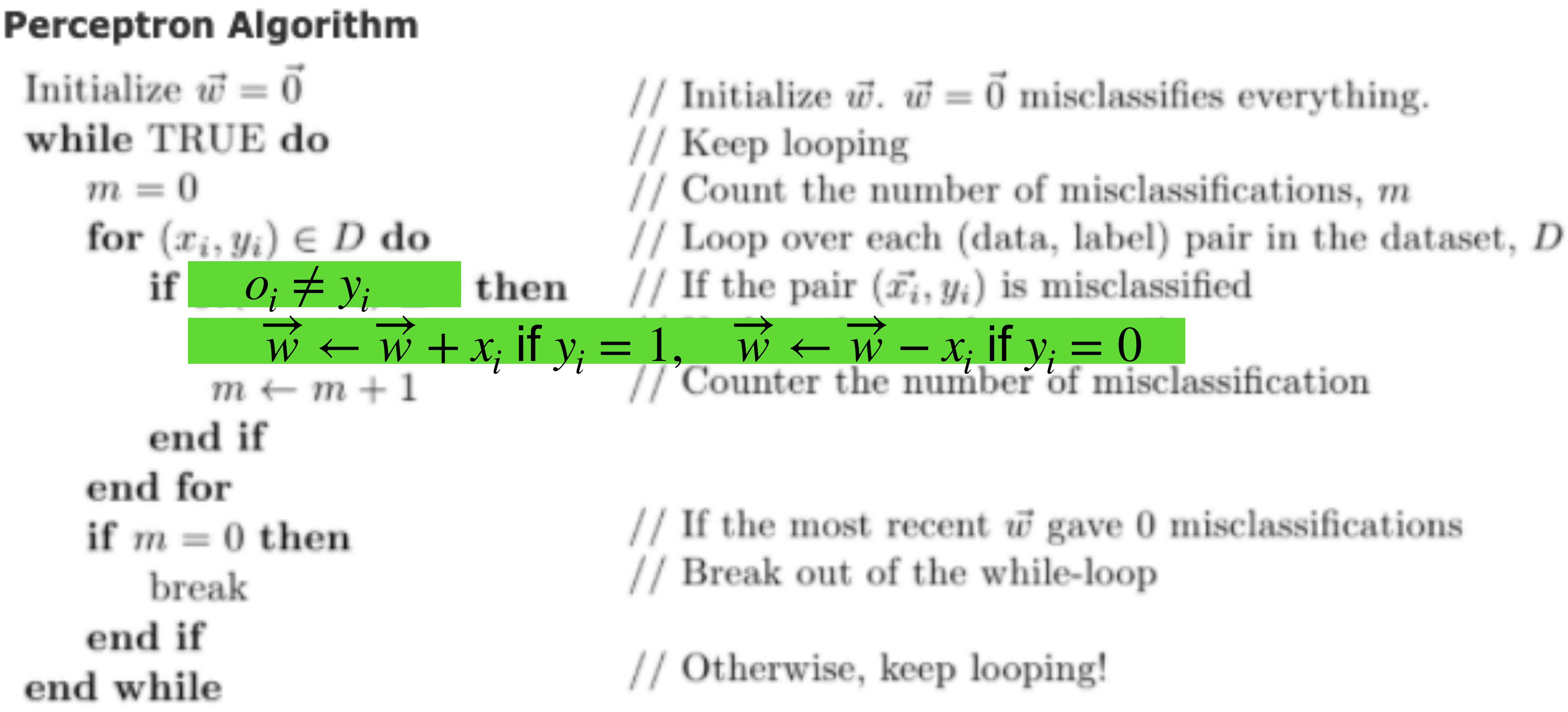
- Goal: learn parameters $\mathbf{w} = \{w_1, w_2, \ldots, w_d\}$ and b to minimize the classification error

Cats vs. dogs?



Training the Perceptron

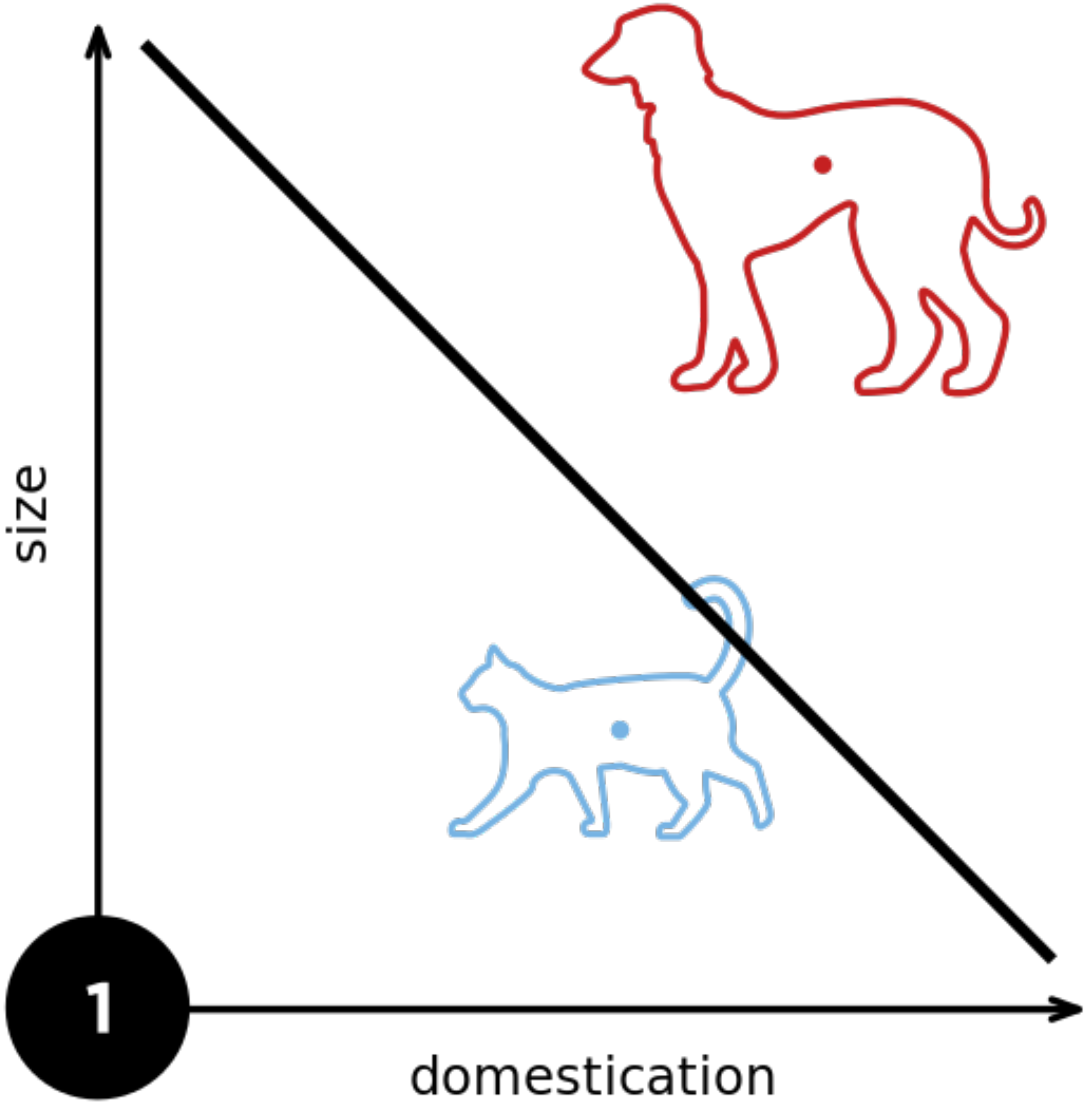
x augmented with dimension of constant 1



Perceptron Algorithm

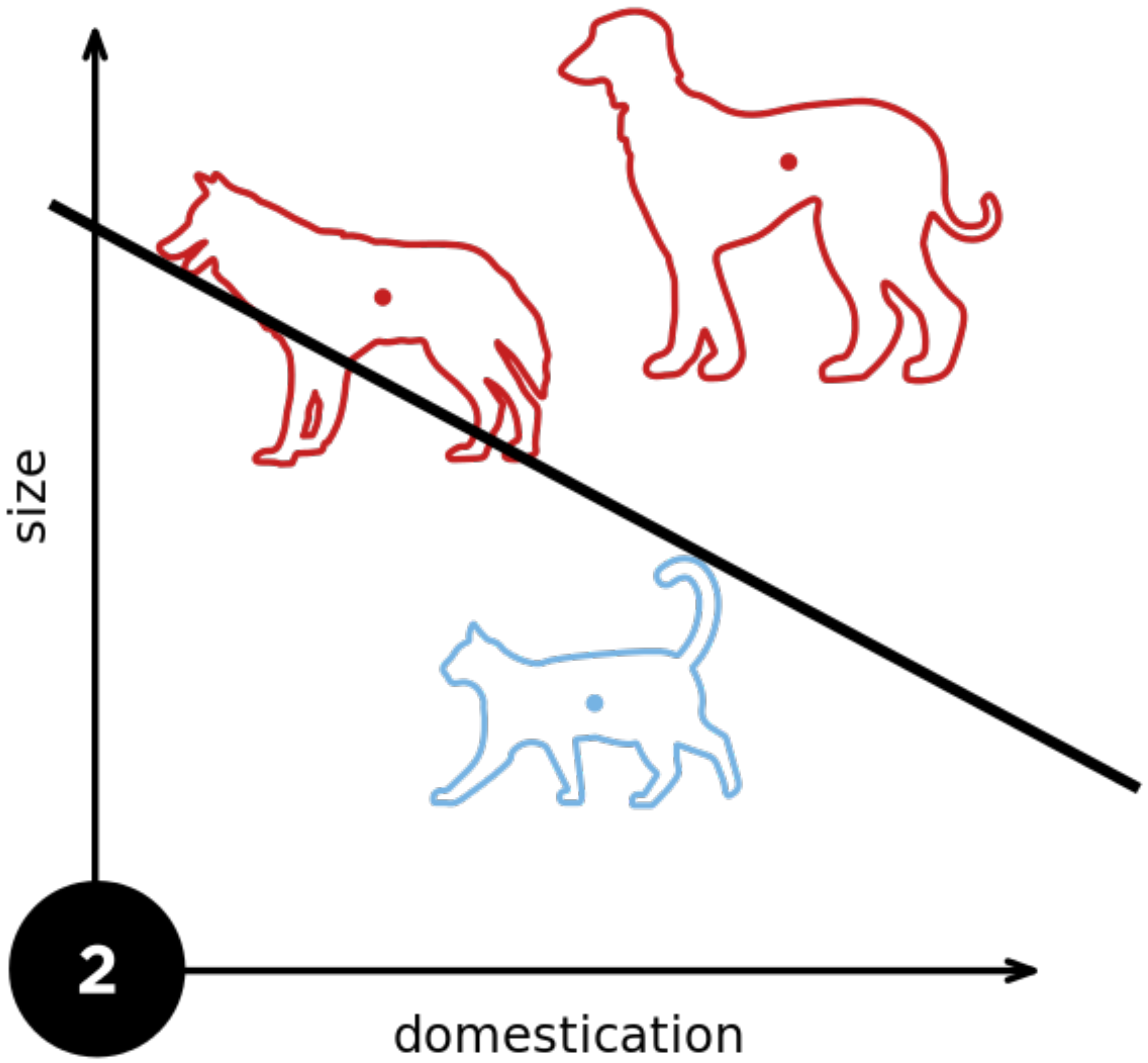
```
Initialize $\vec{w} = \vec{0}$                      // Initialize $\vec{w}$. $\vec{w} = \vec{0}$ misclassifies everything.
while TRUE do                                      // Keep looping
    $m = 0$                                        // Count the number of misclassifications, $m$
    for $(x_i, y_i) \in D$ do                      // Loop over each (data, label) pair in the dataset, $D$
        if $o_i \neq y_i$ then                     // If the pair $(\vec{x}_i, y_i)$ is misclassified
            $\vec{w} \leftarrow \vec{w} + x_i$ if $y_i = 1$,   $\vec{w} \leftarrow \vec{w} - x_i$ if $y_i = 0$
            $m \leftarrow m + 1$                   // Counter the number of misclassification
        end if
    end for
    if $m = 0$ then                                // If the most recent $\vec{w}$ gave 0 misclassifications
        break                                      // Break out of the while-loop
    end if
end while                                          // Otherwise, keep looping!
```

Perceptron



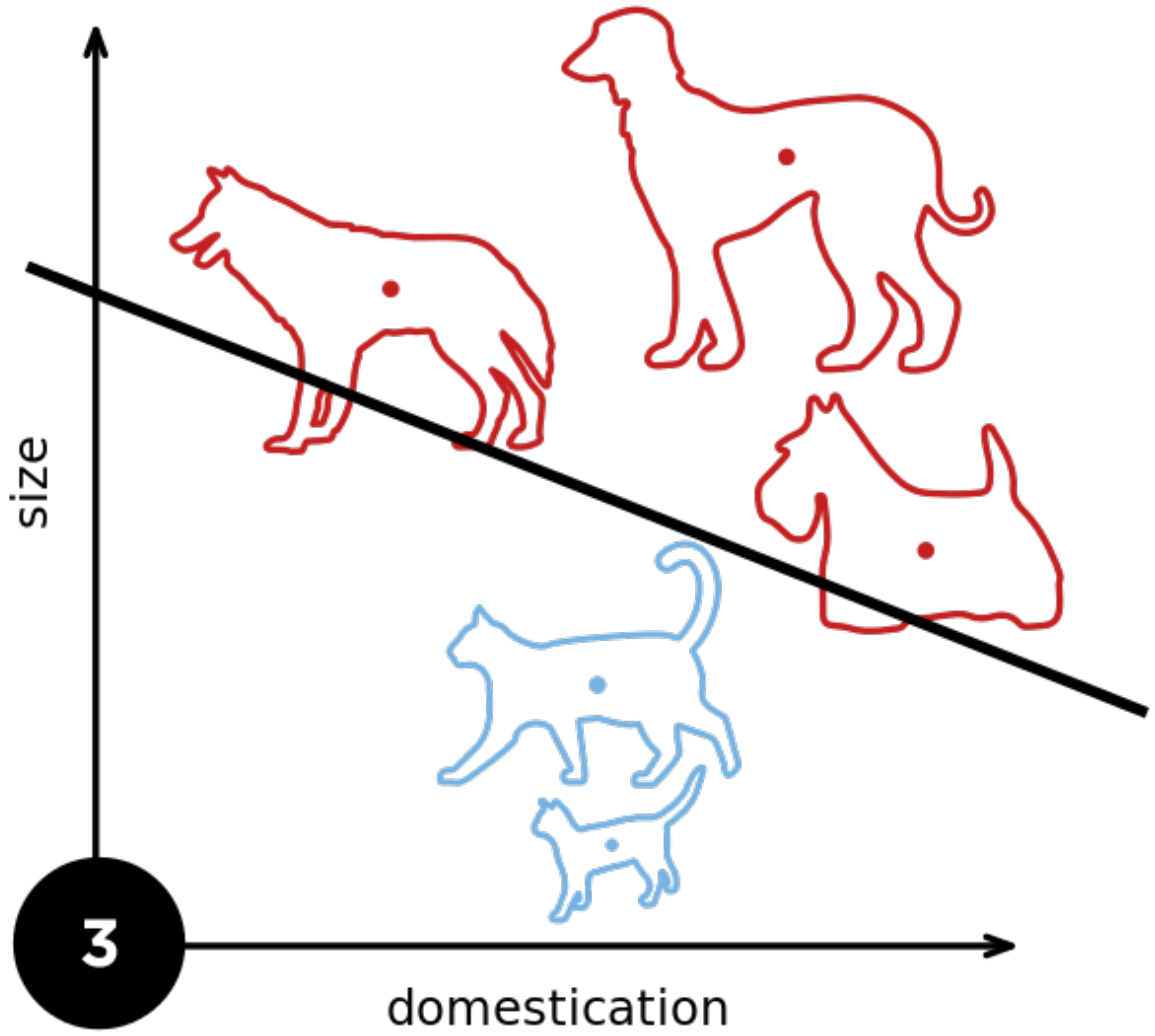
From wikipedia

Perceptron



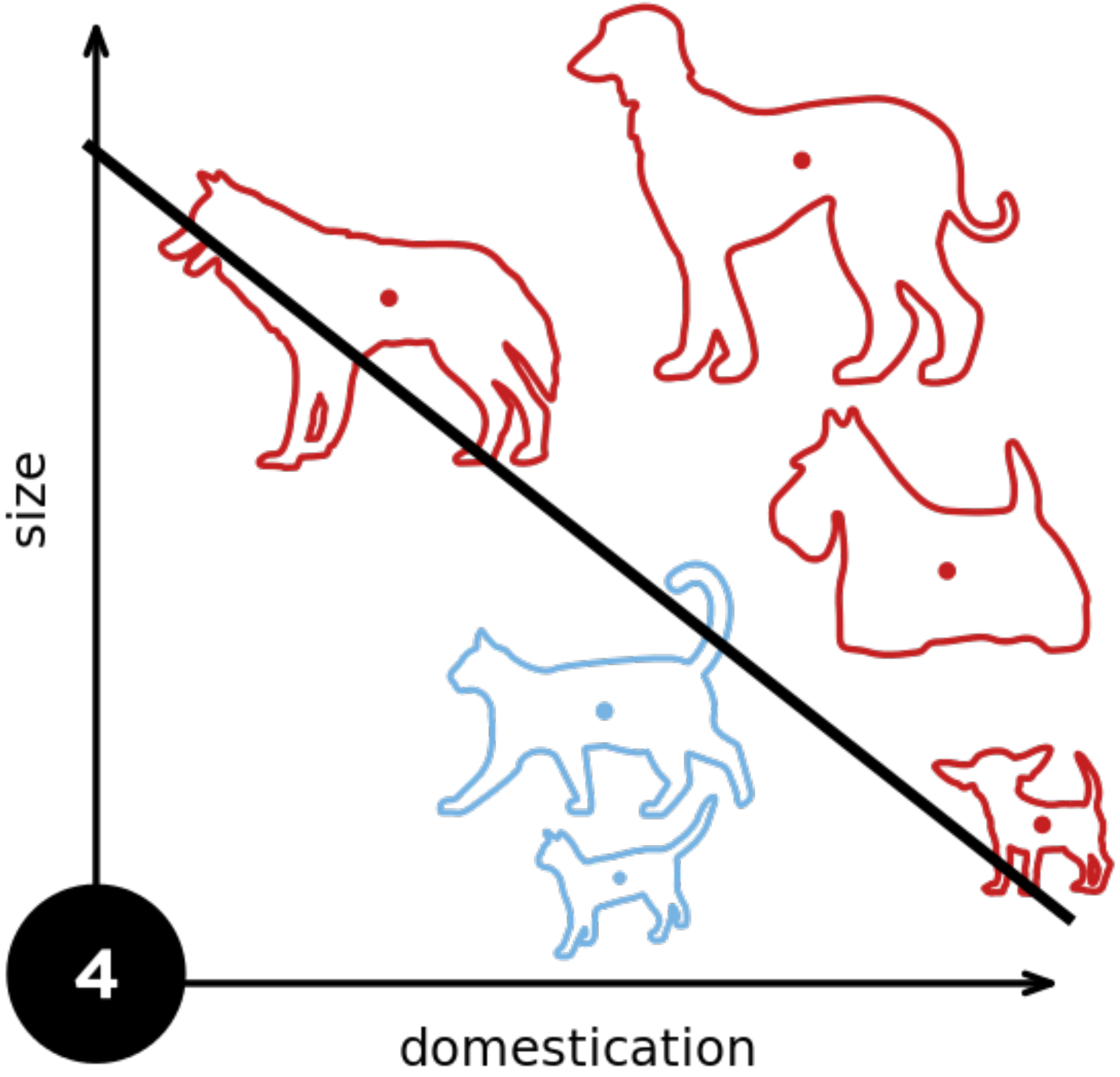
From wikipedia

Perceptron



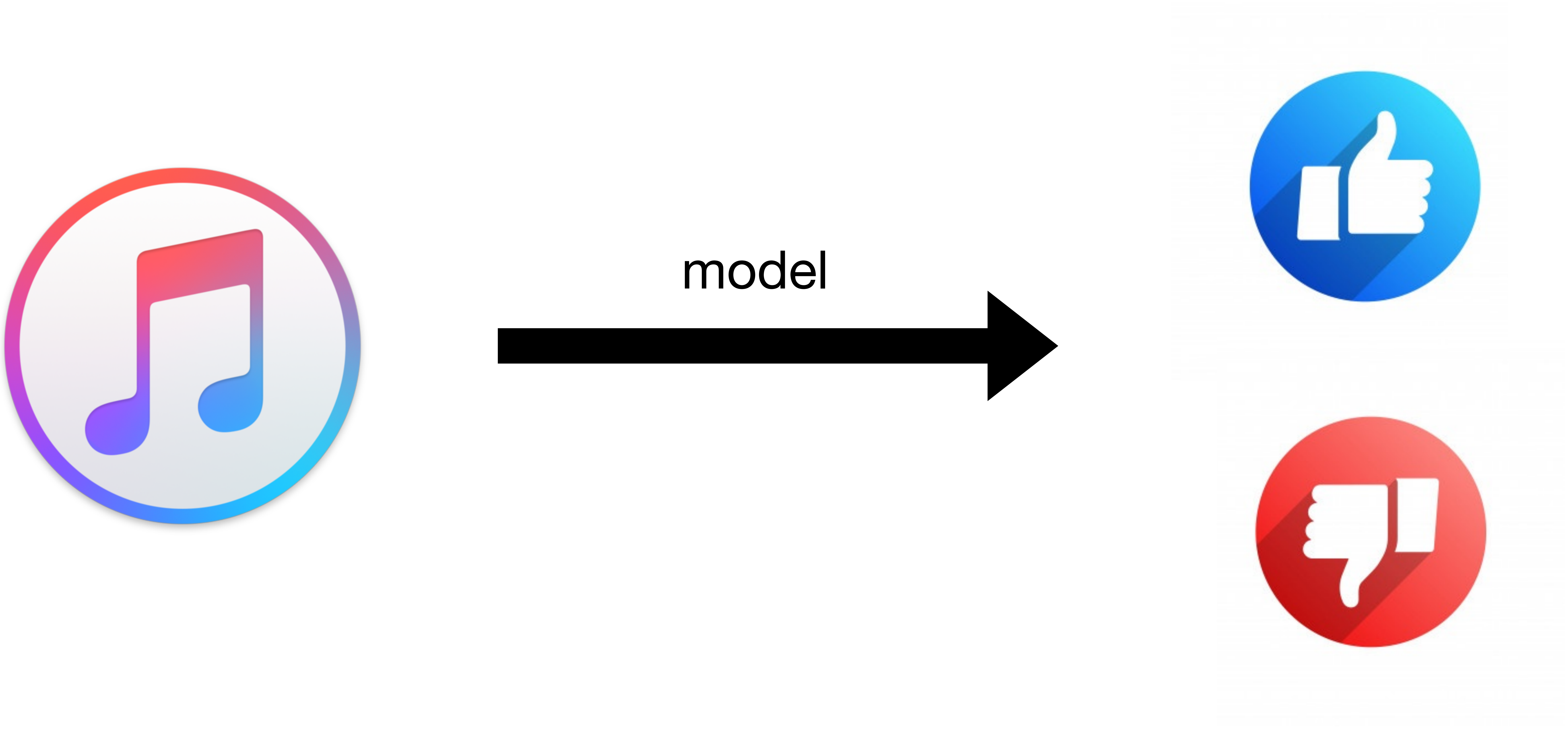
From wikipedia

Perceptron



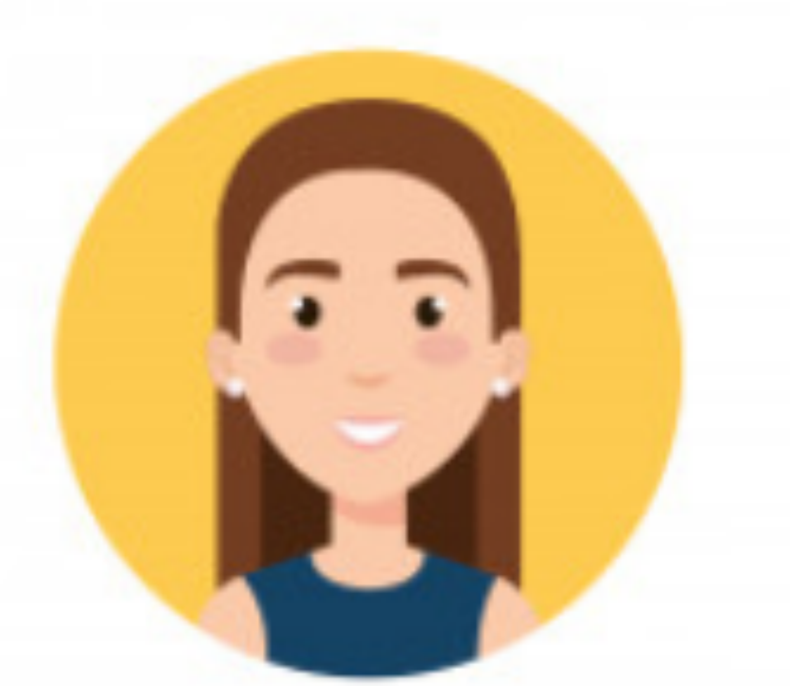
From wikipedia

Example 2: Predict whether a user likes a song or not



model

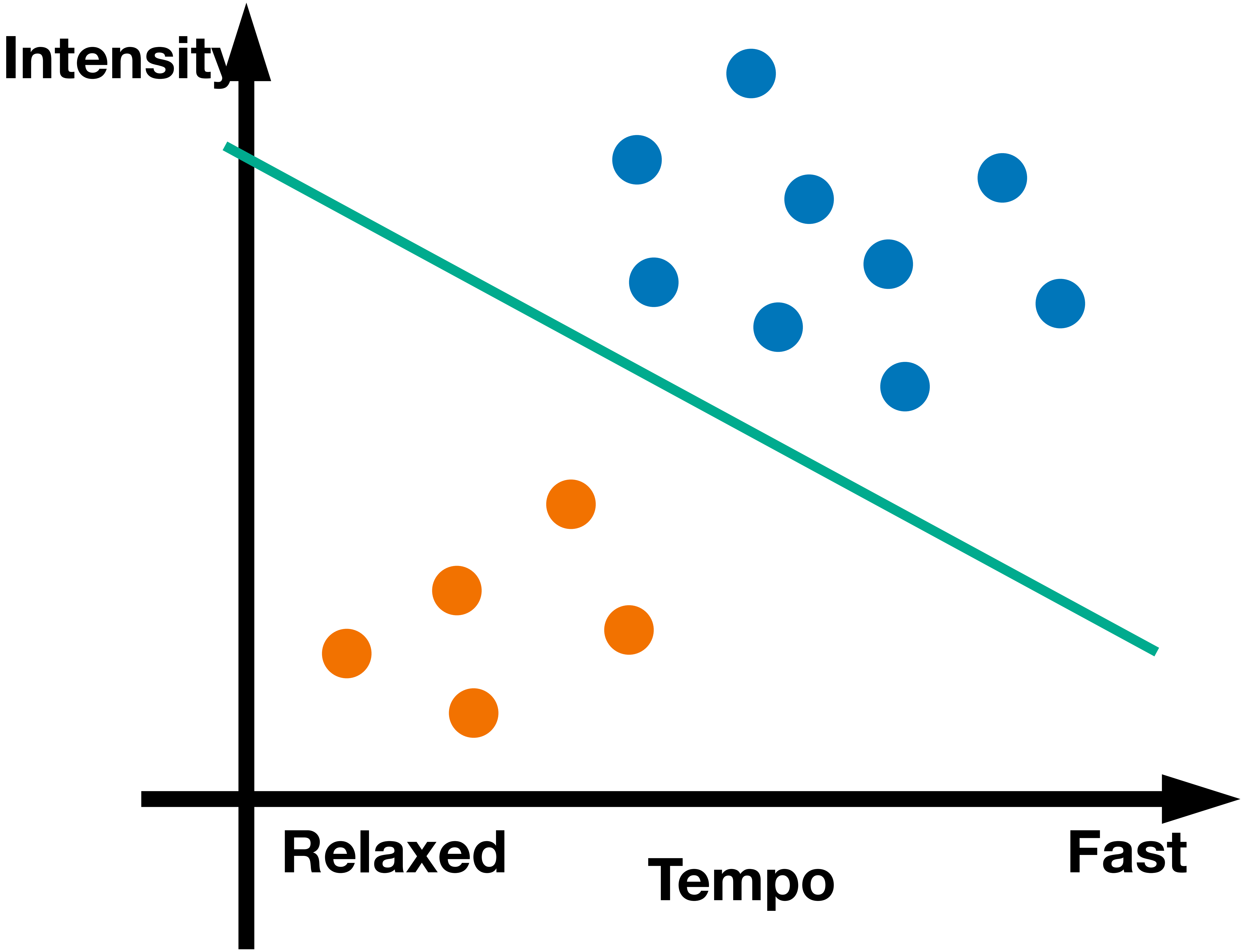
Example 2: Predict whether a user likes a song or not Using Perceptron



User Sharon

● DisLike

● Like



Learning AND function using perceptron

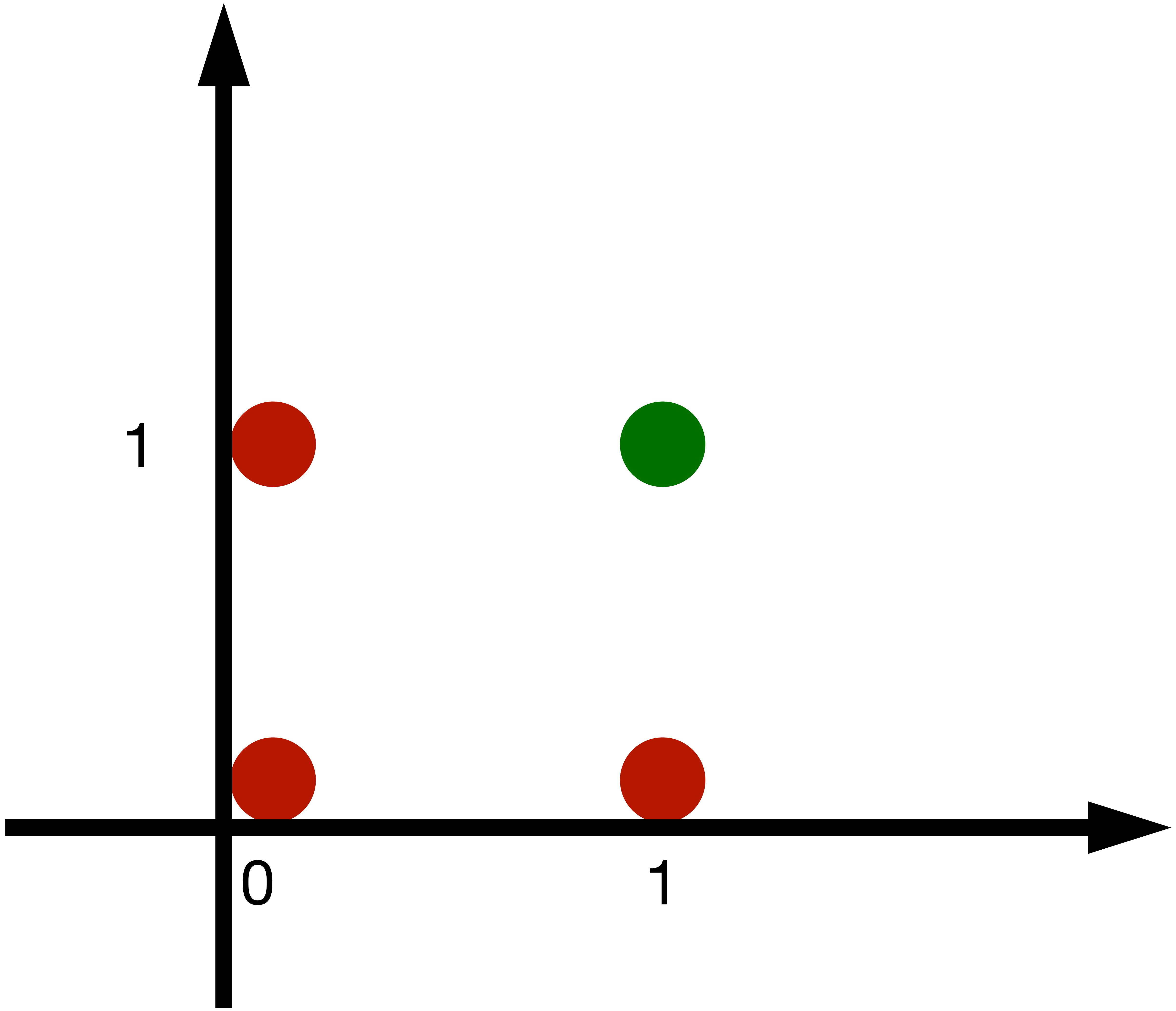
The perceptron can learn an AND function

$x_1 = 1, x_2 = 1, y = 1$

$x_1 = 1, x_2 = 0, y = 0$

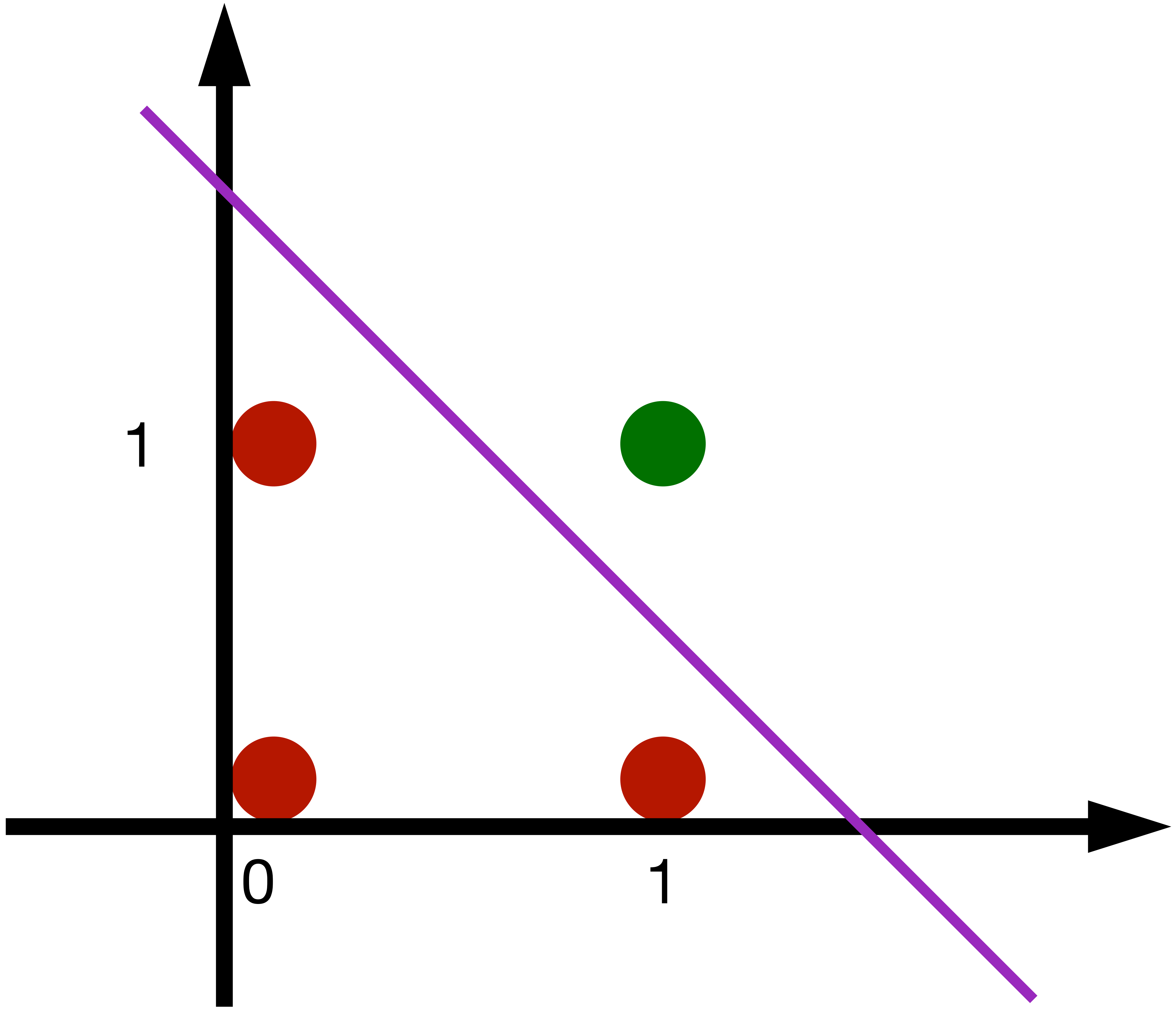
$x_1 = 0, x_2 = 1, y = 0$

$x_1 = 0, x_2 = 0, y = 0$



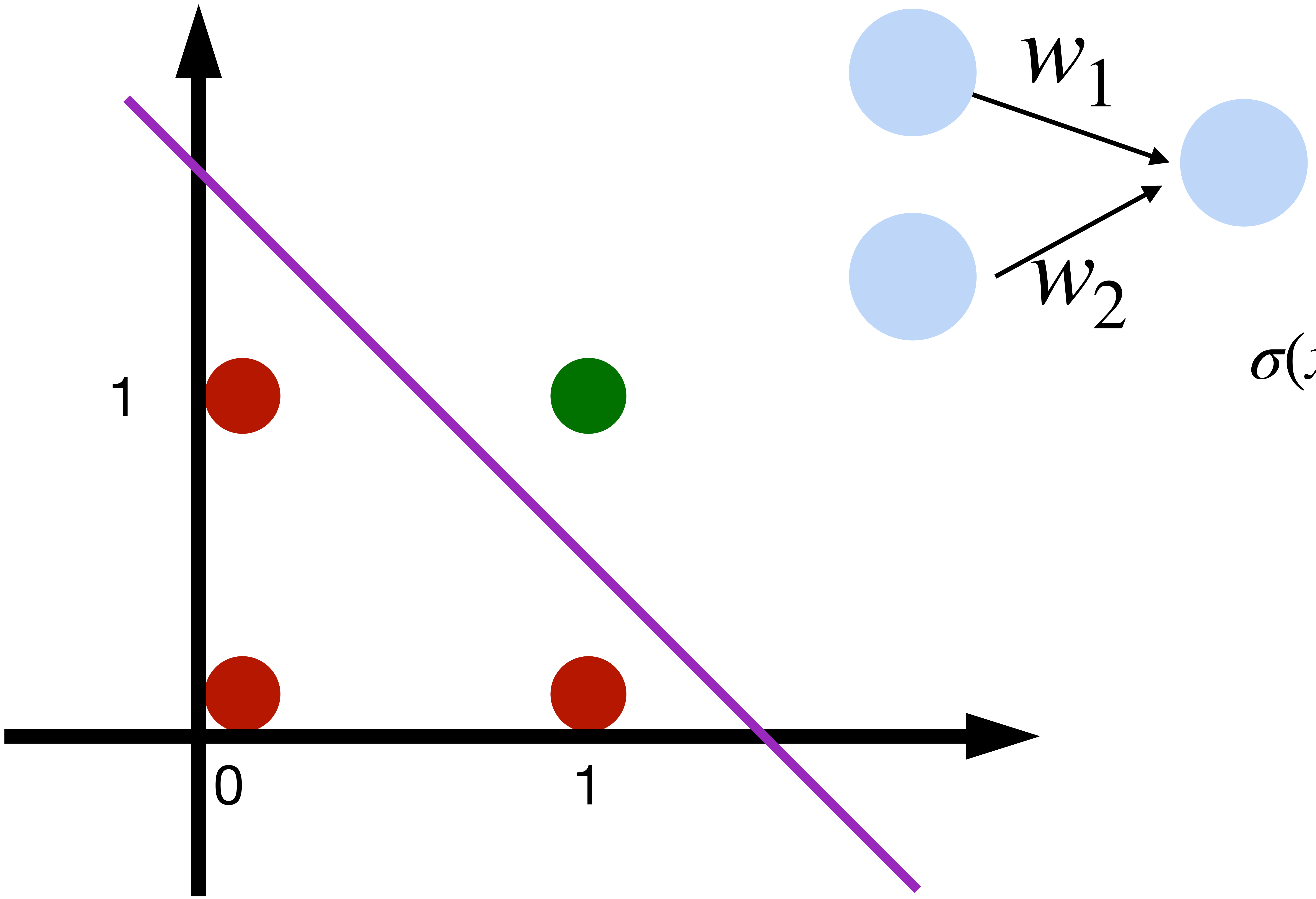
Learning AND function using perceptron

The perceptron can learn an AND function



Learning AND function using perceptron

The perceptron can learn an AND function

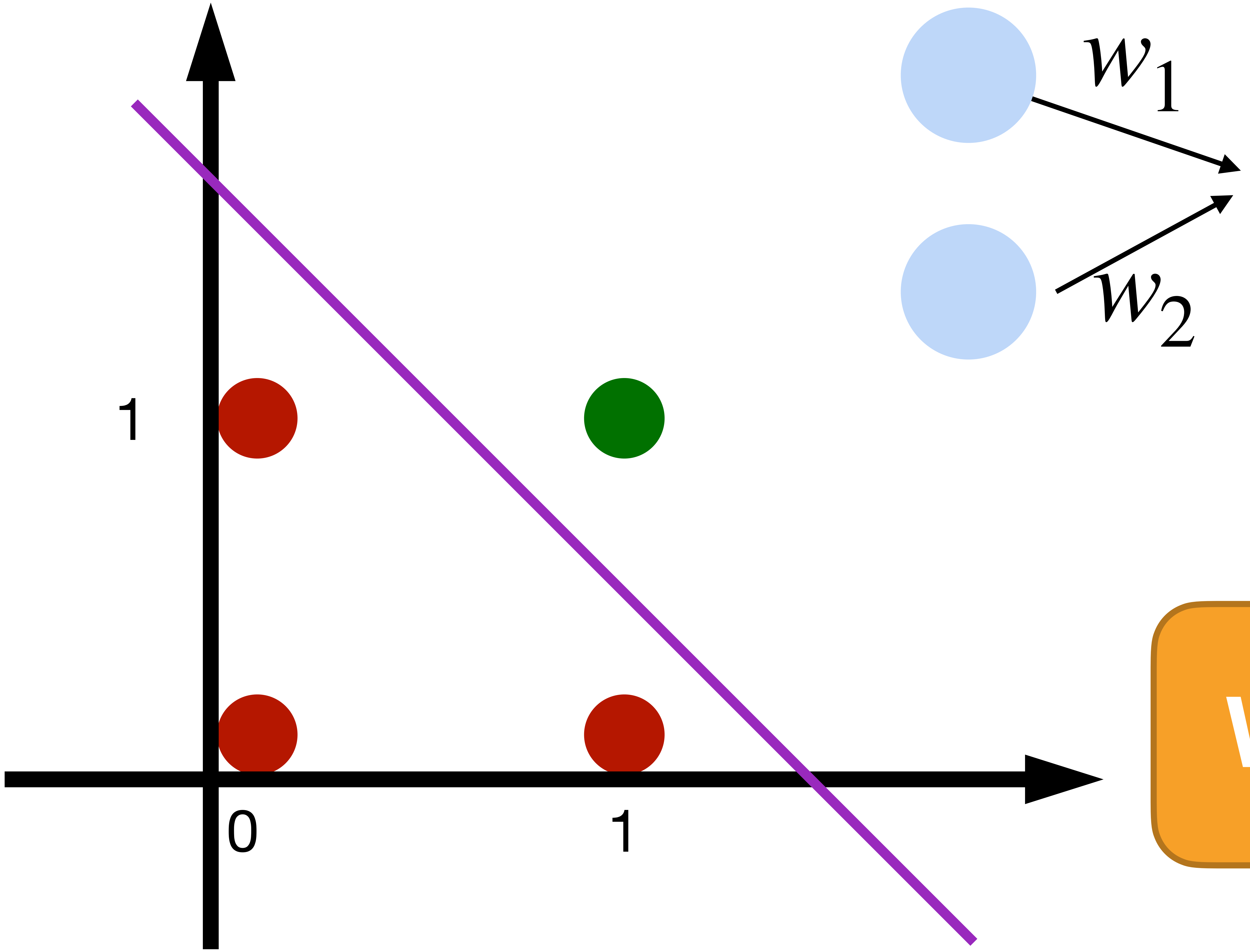


Output $\sigma(x_1 w_1 + x_2 w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Learning AND function using perceptron

The perceptron can learn an AND function



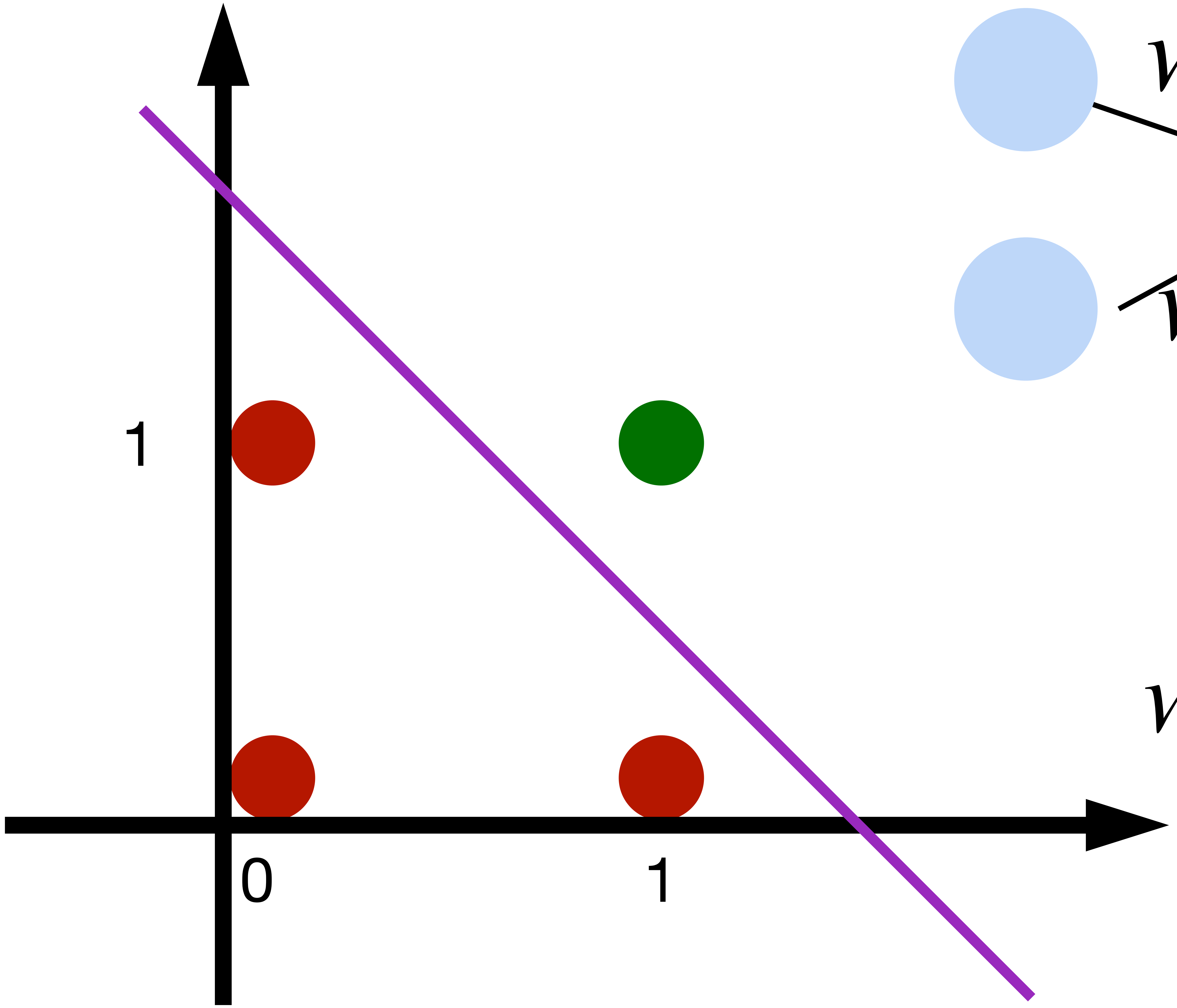
Output $\sigma(x_1 w_1 + x_2 w_2 + b)$

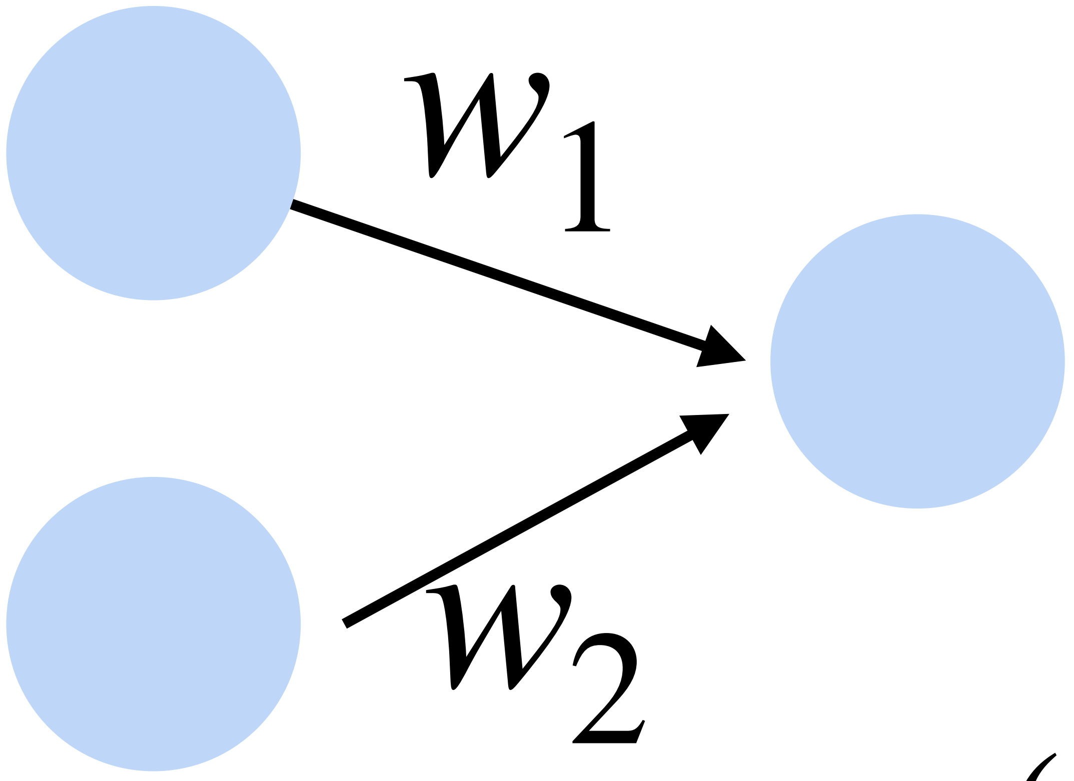
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

What's w and b?

Learning AND function using perceptron

The perceptron can learn an AND function





Output $\sigma(x_1w_1 + x_2w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 1, w_2 = 1, b = -1.5$$

Learning OR function using perceptron

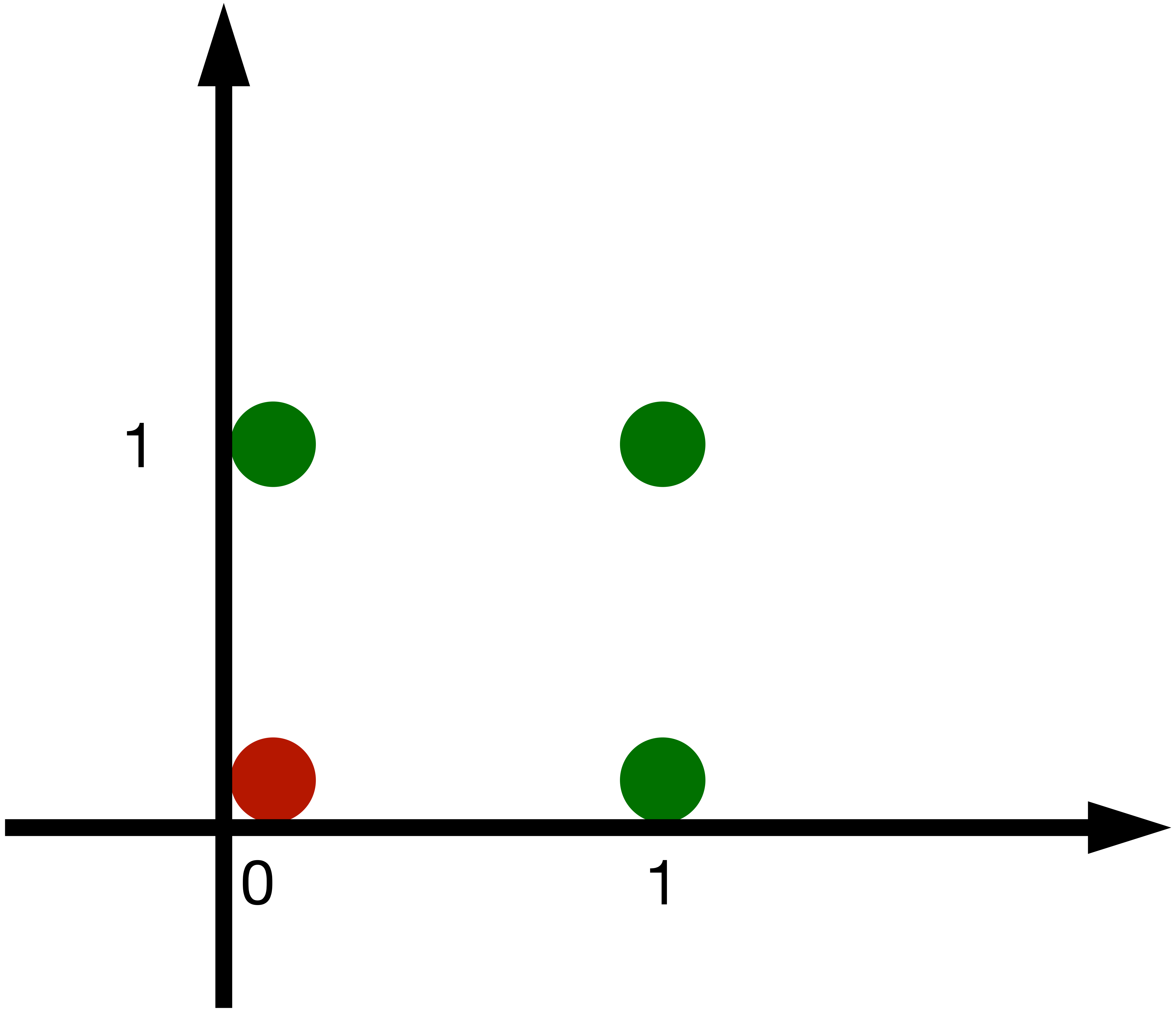
The perceptron can learn an OR function

$$x_1 = 1, x_2 = 1, y = 1$$

$$x_1 = 1, x_2 = 0, y = 1$$

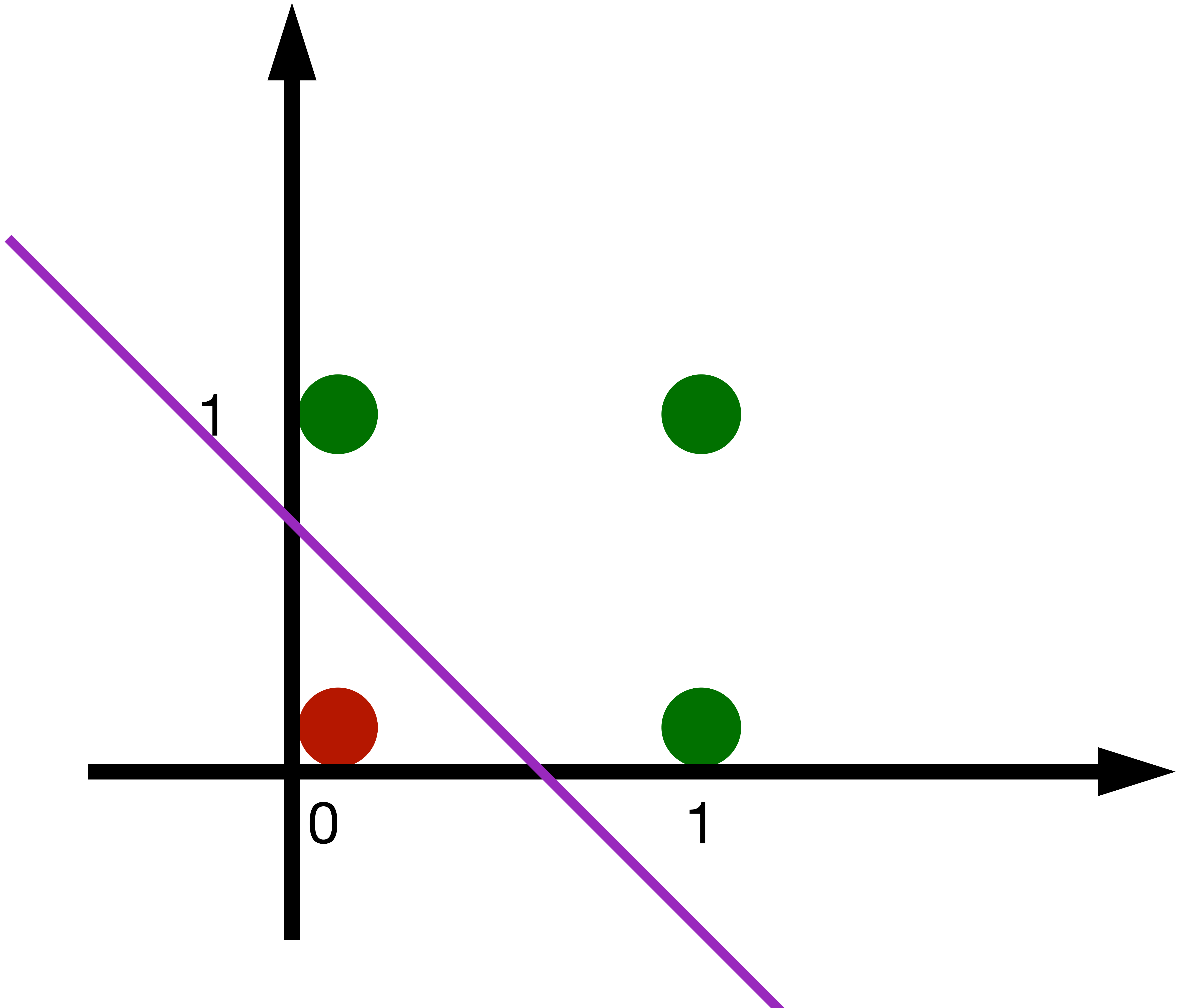
$$x_1 = 0, x_2 = 1, y = 1$$

$$x_1 = 0, x_2 = 0, y = 0$$



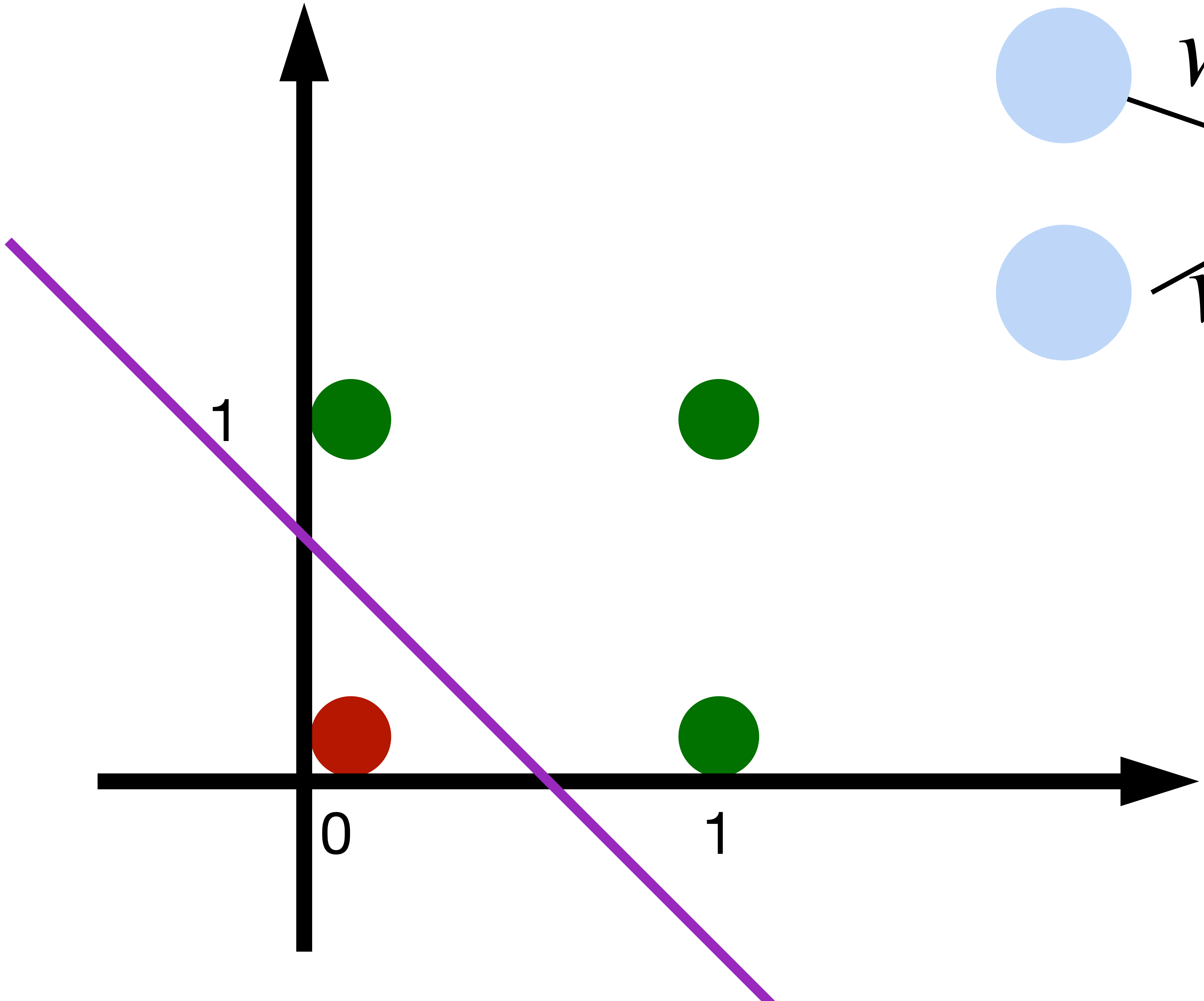
Learning OR function using perceptron

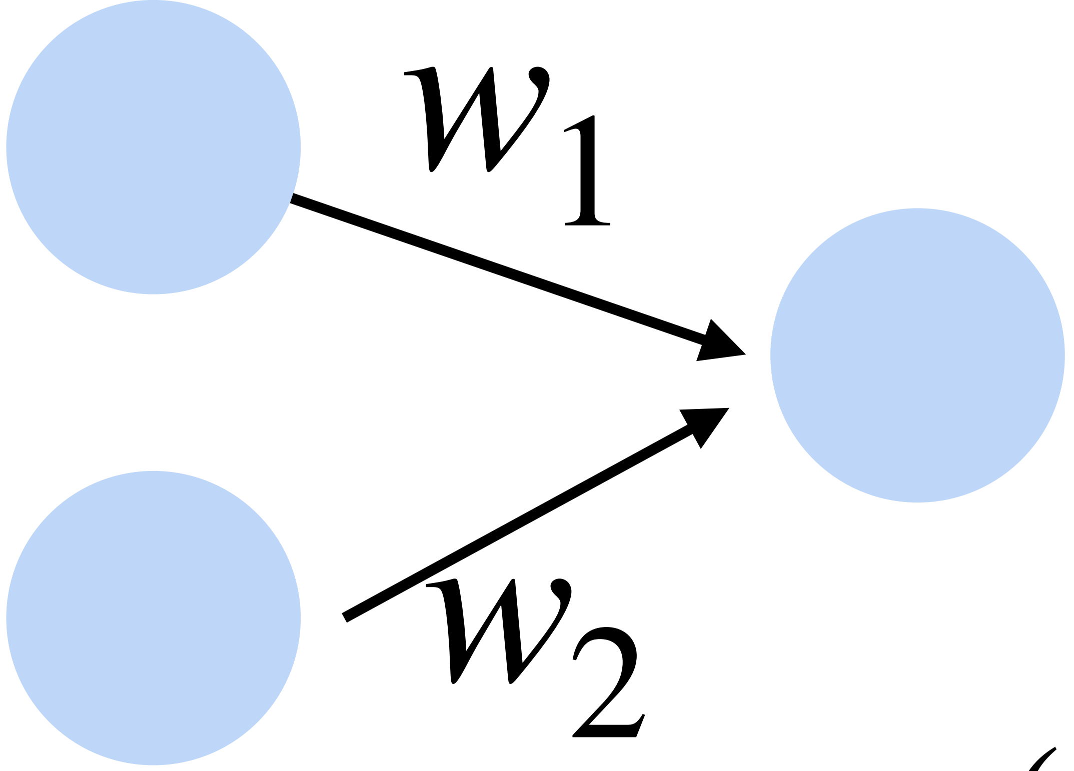
The perceptron can learn an OR function



Learning OR function using perceptron

The perceptron can learn an OR function



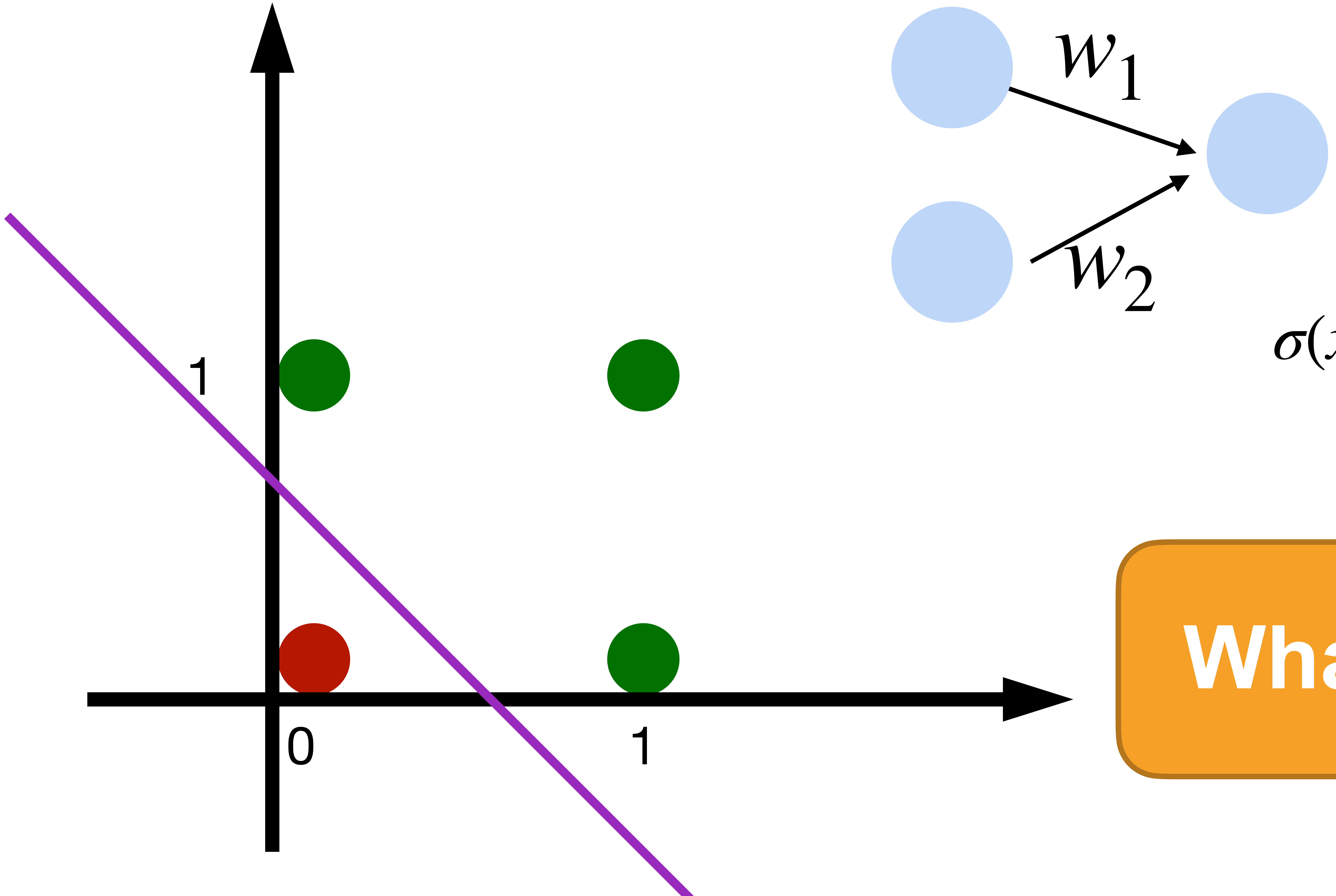


Output $\sigma(x_1 w_1 + x_2 w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Learning OR function using perceptron

The perceptron can learn an OR function



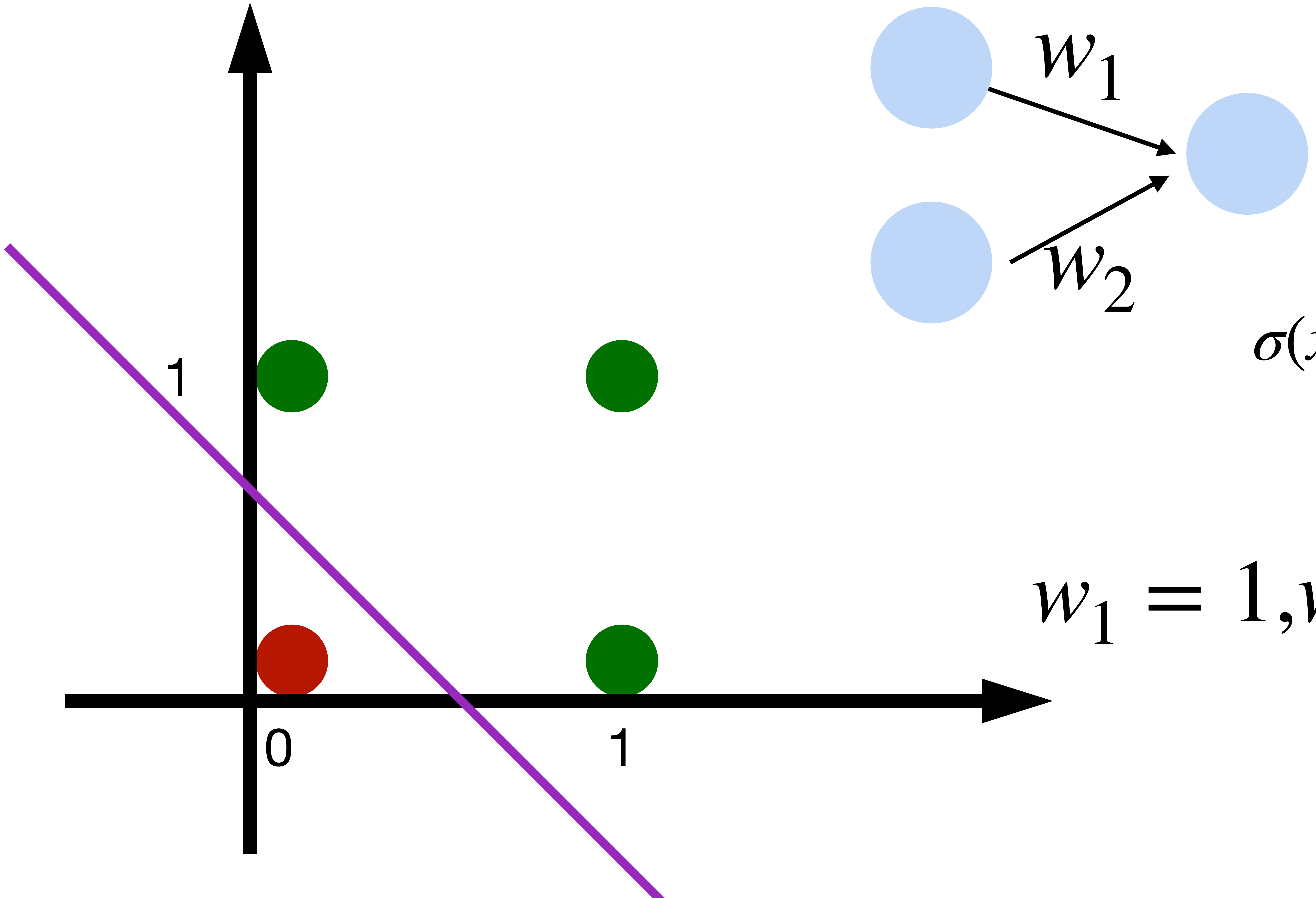
Output $\sigma(x_1 w_1 + x_2 w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

What’s w and b?

Learning OR function using perceptron

The perceptron can learn an OR function



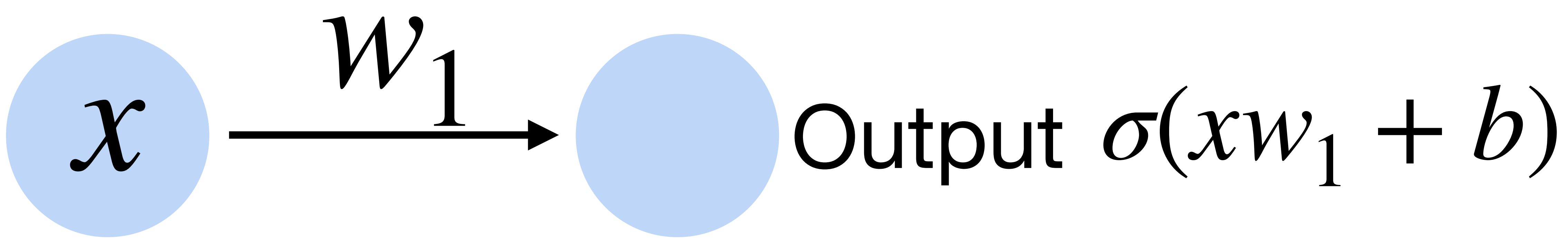
Output $\sigma(x_1w_1 + x_2w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

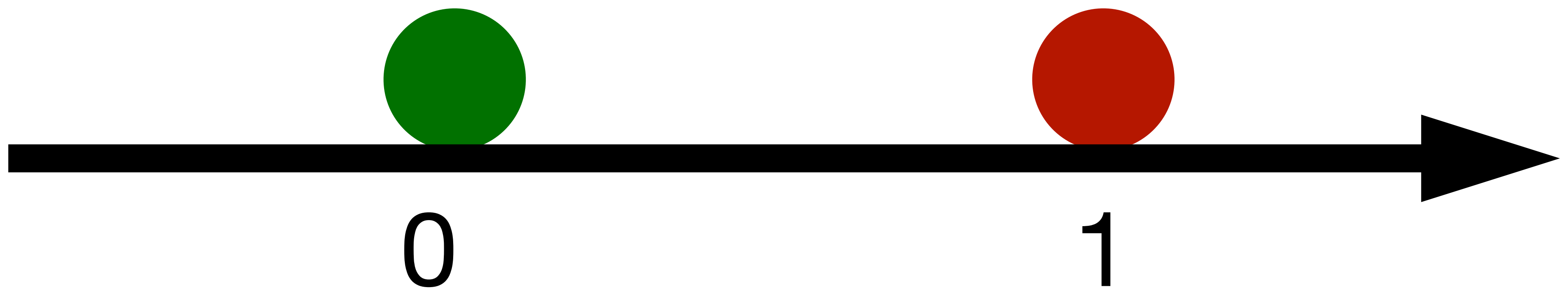
$$w_1 = 1, w_2 = 1, b = -0.5$$

Learning NOT function using perceptron

The perceptron can learn NOT function (single input)

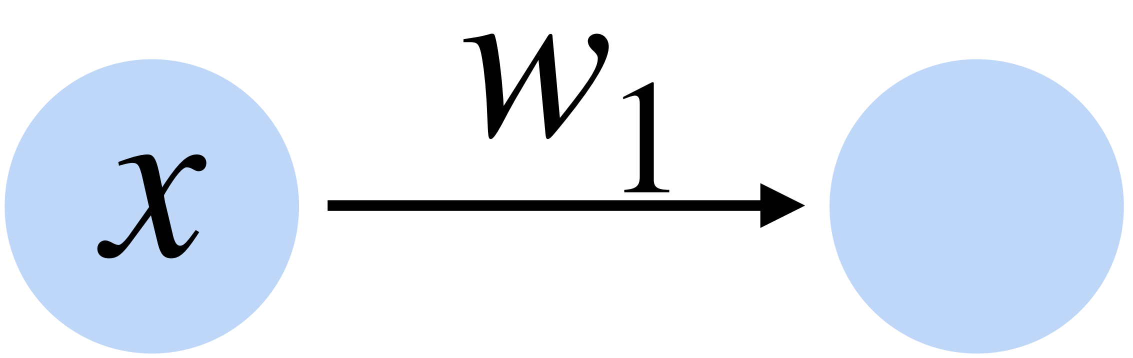


$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



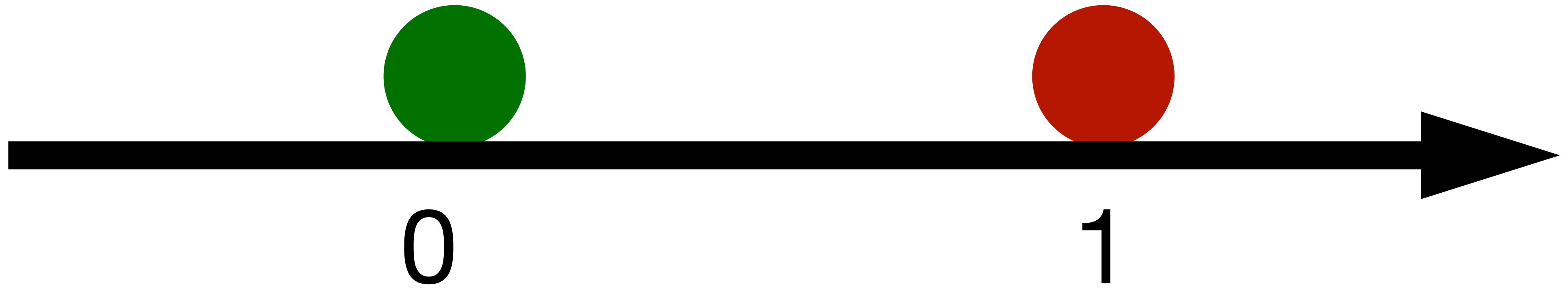
Learning NOT function using perceptron

The perceptron can learn NOT function (single input)



Output $\sigma(xw_1 + b)$

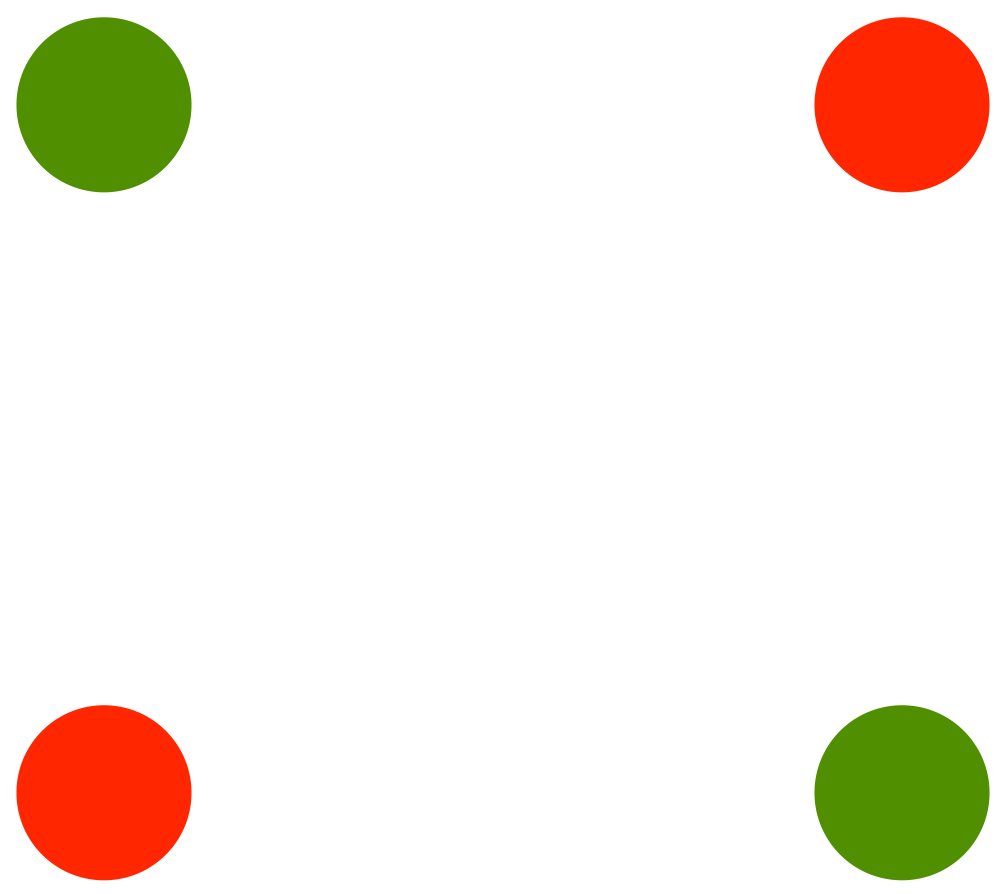
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



$$w_1 = -1, b = 0.5$$

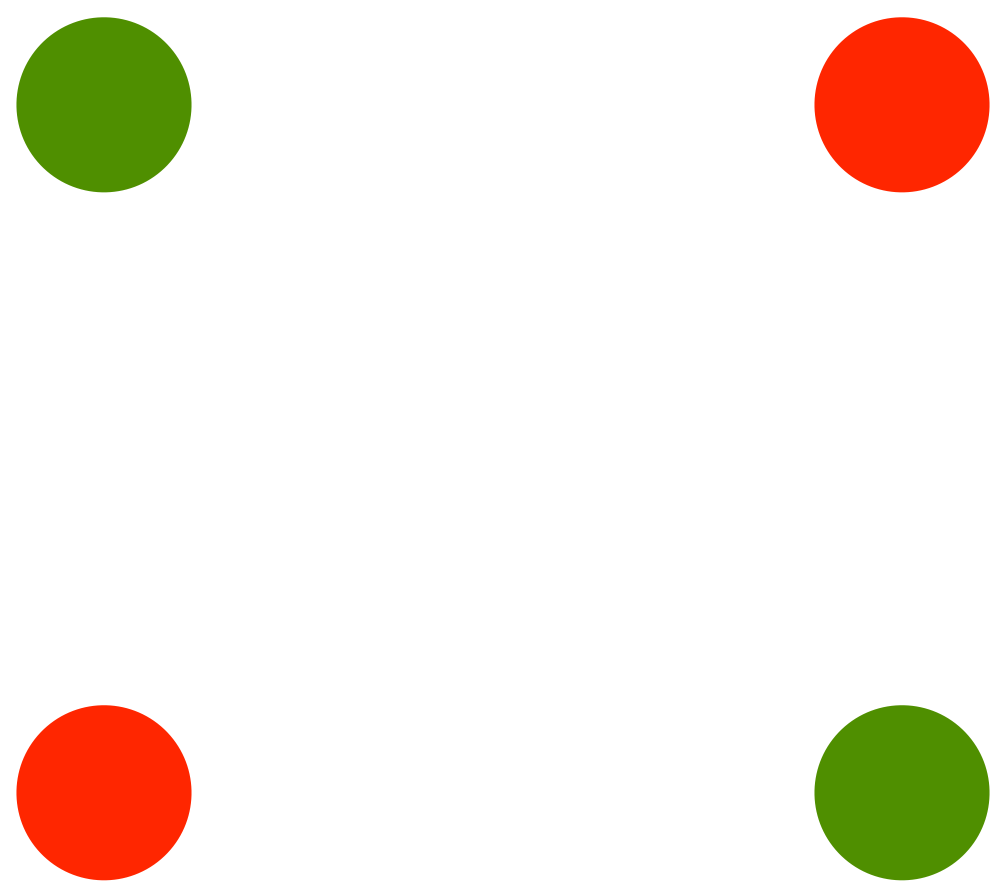
XOR Problem (Minsky & Papert, 1969)

The perceptron cannot learn an XOR function
(it can only generate linear separators)



XOR Problem (Minsky & Papert, 1969)

The perceptron cannot learn an XOR function
(it can only generate linear separators)



This contributed to the first AI winter

Quiz Break

Perceptron can be used for representing:

A. AND function

B. OR function

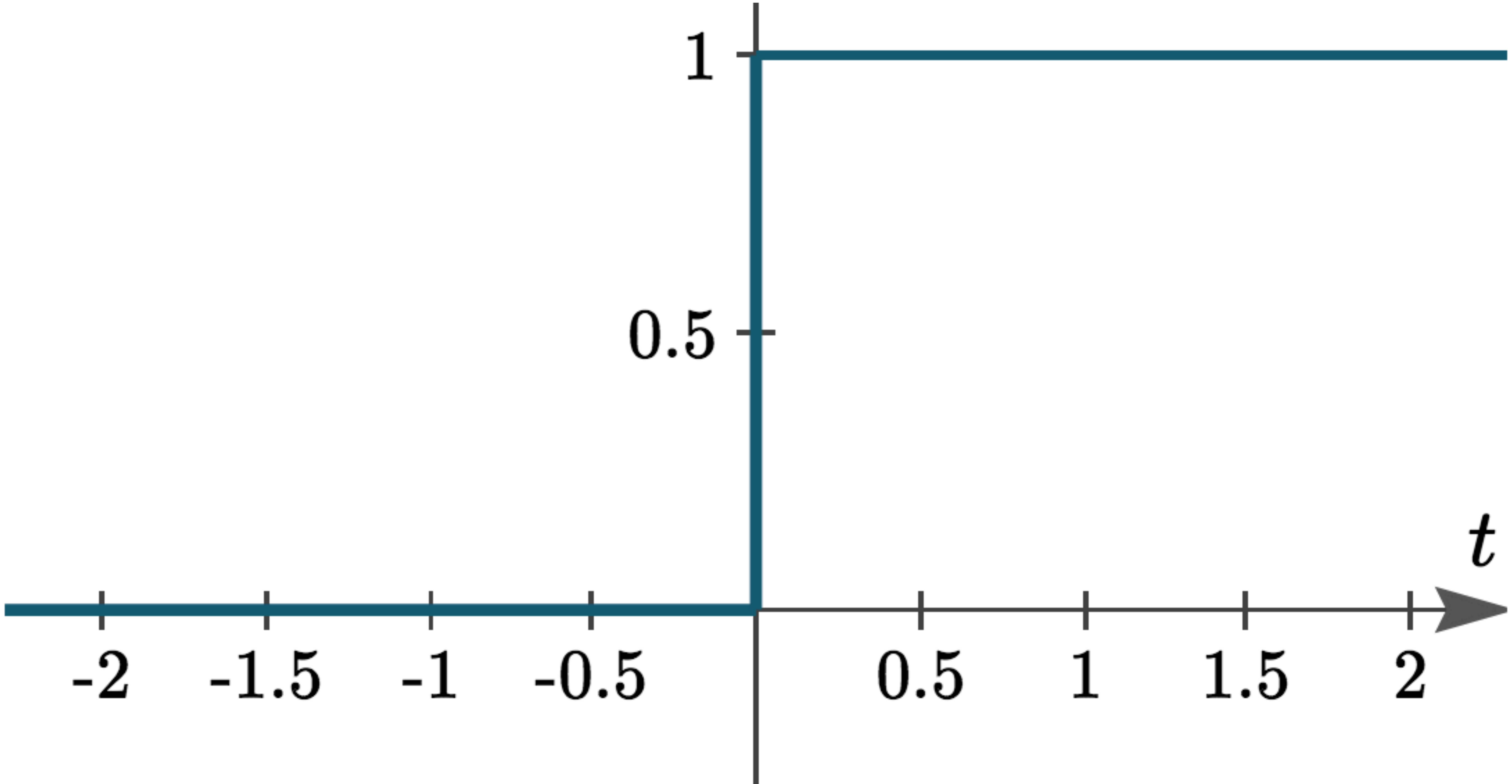
C. XOR function

D. Both AND and OR function

Step Function activation

Step function is discontinuous, which cannot be used for gradient descent

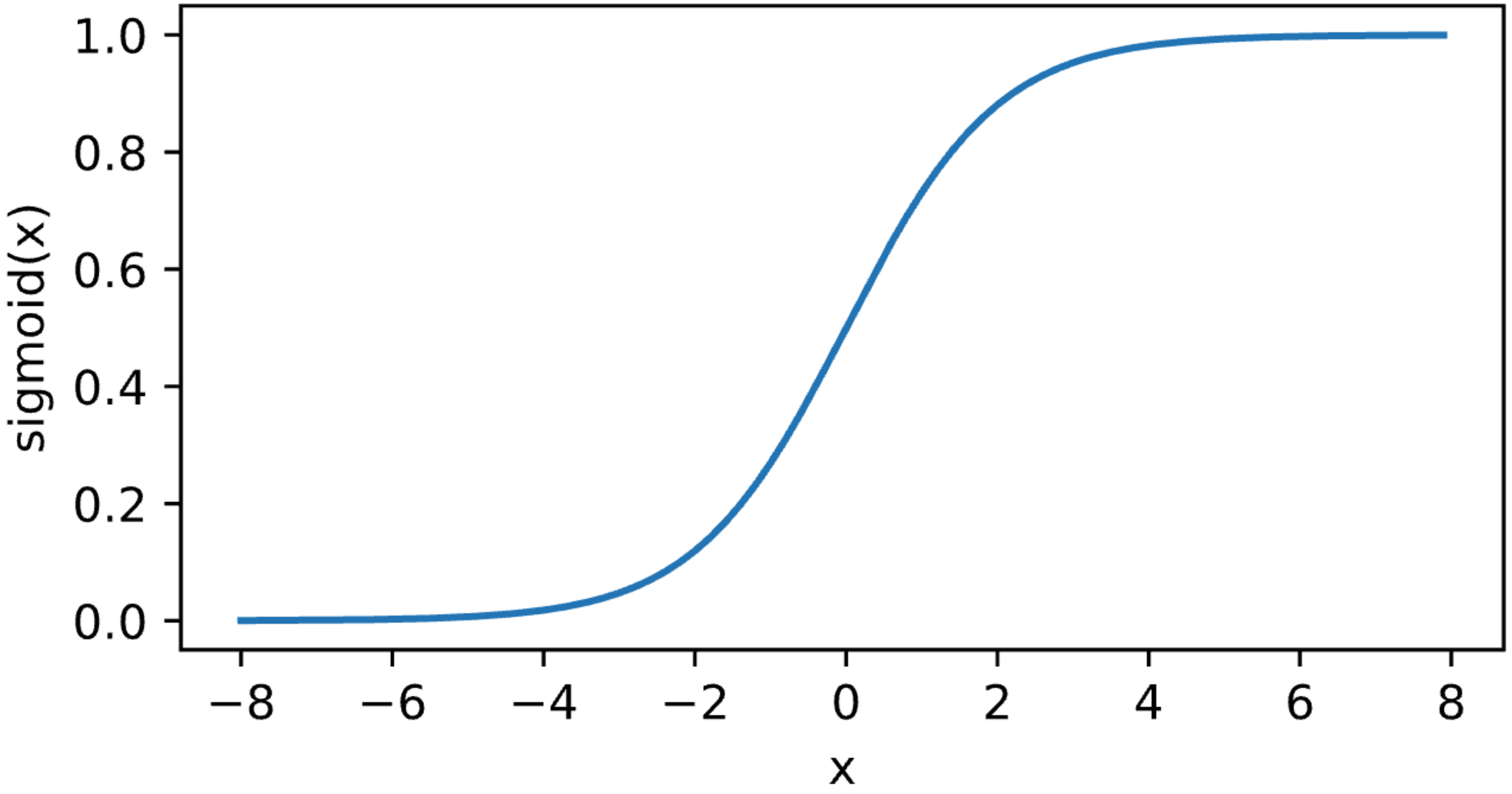
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



Sigmoid/Logistic Activation

Map input into [0, 1], a **soft** version of $\sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$

$$\sigma(z) = \text{sigmoid}(z) = \frac{1}{1 + \exp(-z)}$$

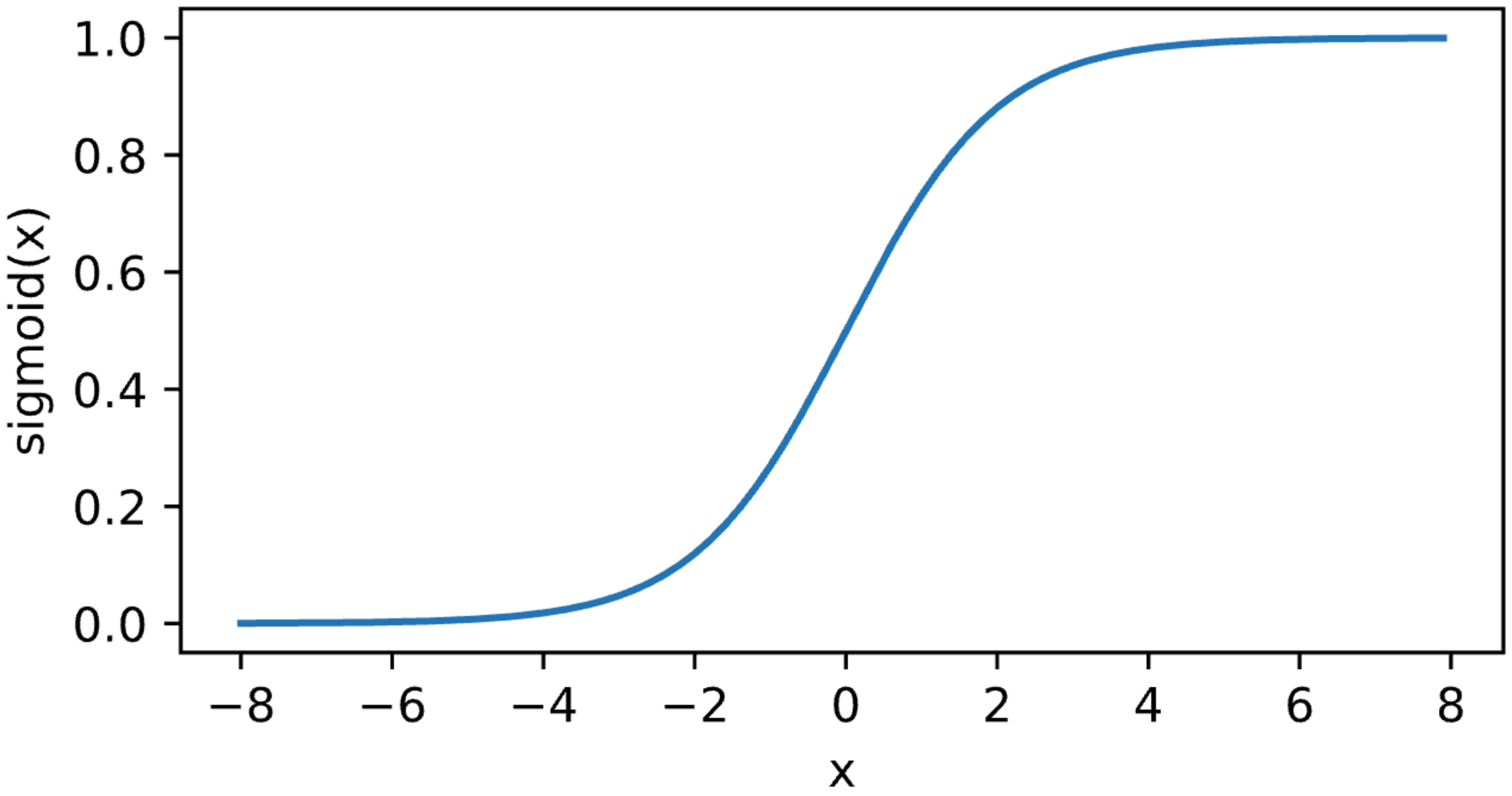


Logistic regression

$\mathbf{x} \in \mathbb{R}^d, y = \{-1, +1\}$

$$p(y = 1 \,|\, \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

$$p(y = -1 \,|\, \mathbf{x}) = 1 - \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T\mathbf{x})}$$



Logistic regression

Given: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ $\mathbf{x} \in \mathbb{R}^d, y = \{-1, +1\}$

Training: maximize likelihood estimate (on the conditional probability)

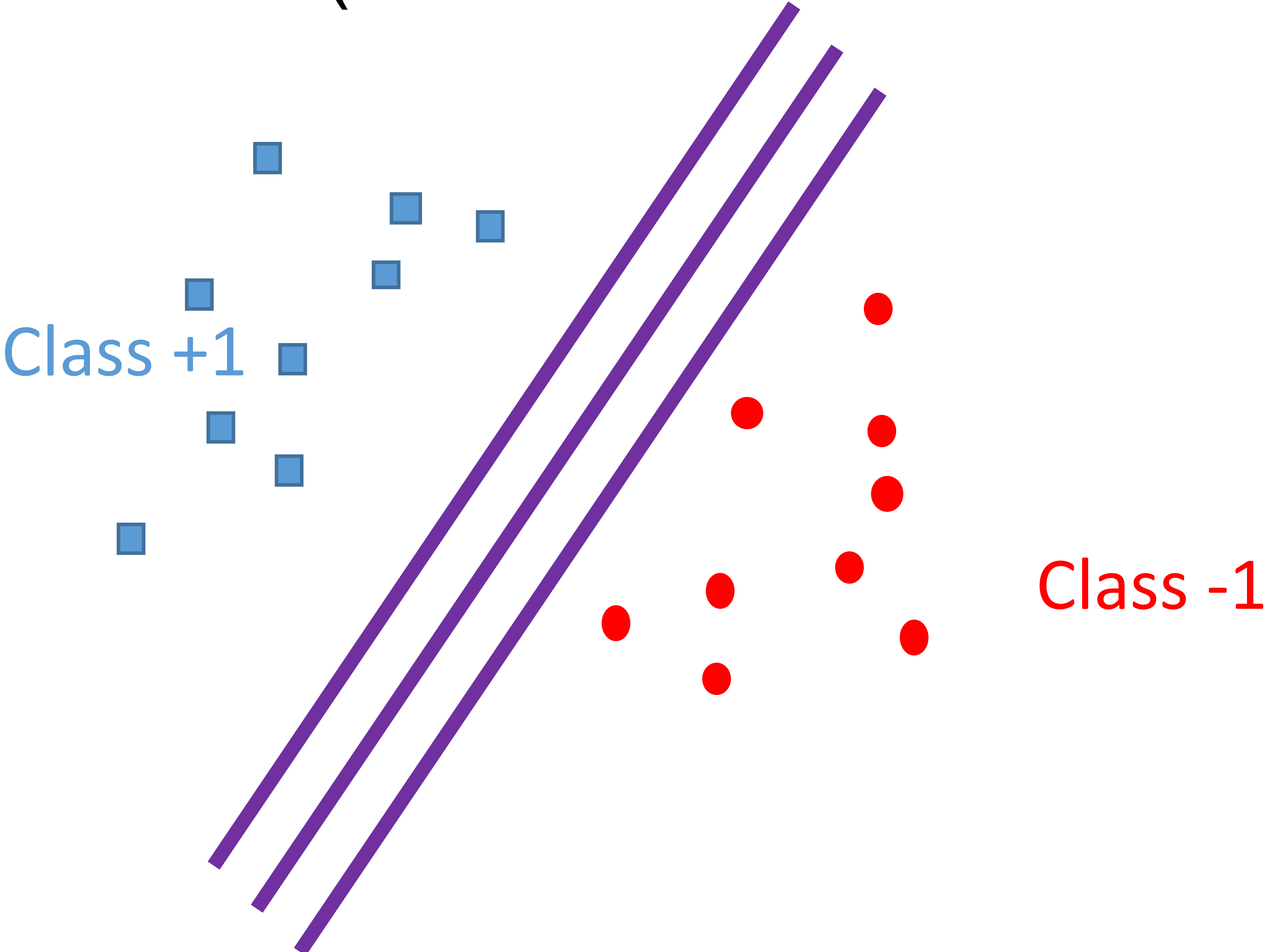
$$\max_{\mathbf{w}} \sum_i \log \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

Logistic regression

Given: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ $\mathbf{x} \in \mathbb{R}^d, y = \{-1, +1\}$

Training: maximize likelihood estimate (on the conditional probability)

When training data is linearly separable, many solutions



Logistic regression

Given: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ $\mathbf{x} \in \mathbb{R}^d, y = \{-1, +1\}$

Training: maximum A posteriori (MAP)

$$\min_{\mathbf{w}} \sum_i -\log \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

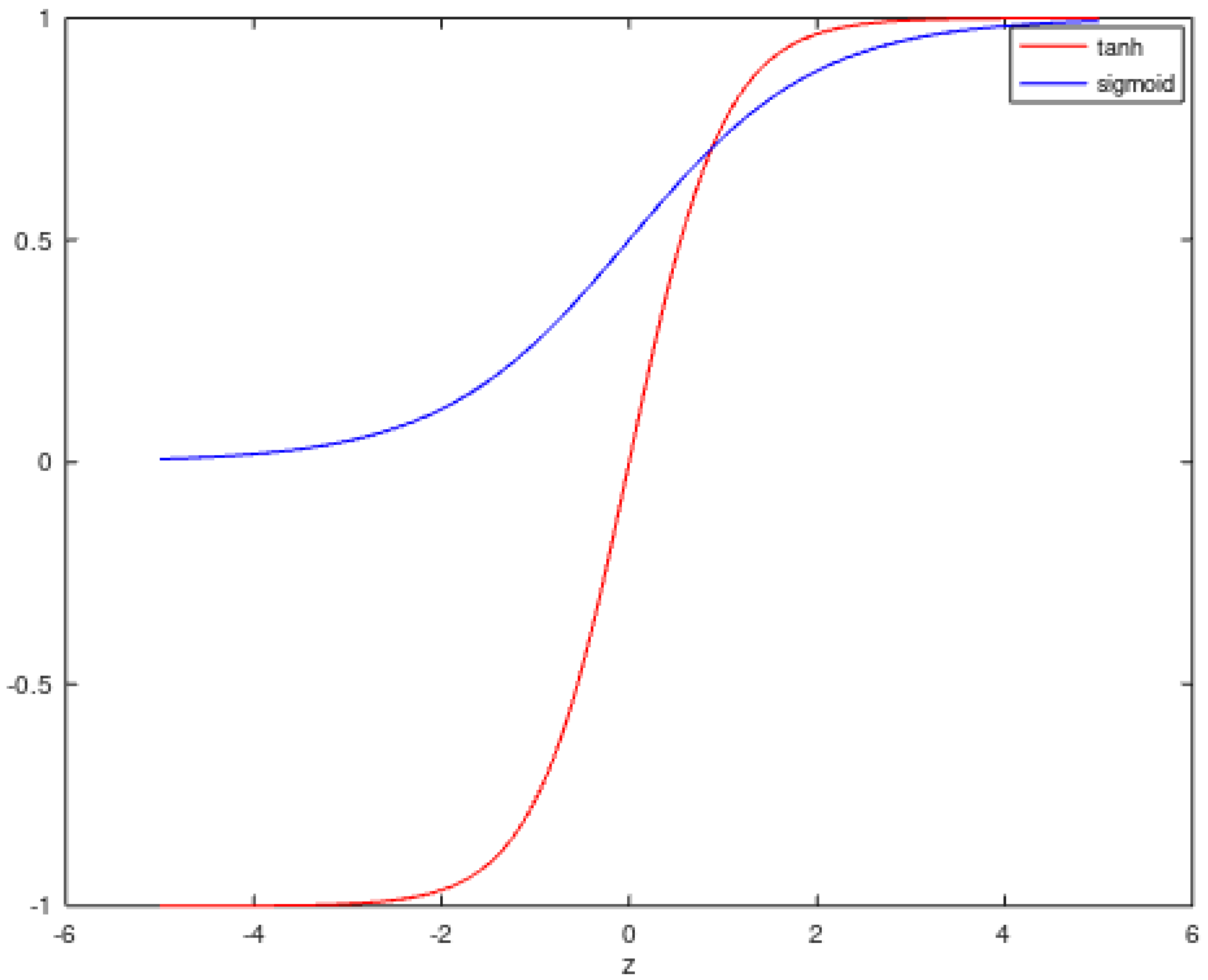
- Convex optimization
- Solve via (stochastic) gradient descent

Tanh Activation

Map inputs into (-1, 1)

$$\sigma(z) = \tanh(z) = \frac{1 - \exp(-2z)}{1 + \exp(-2z)}$$

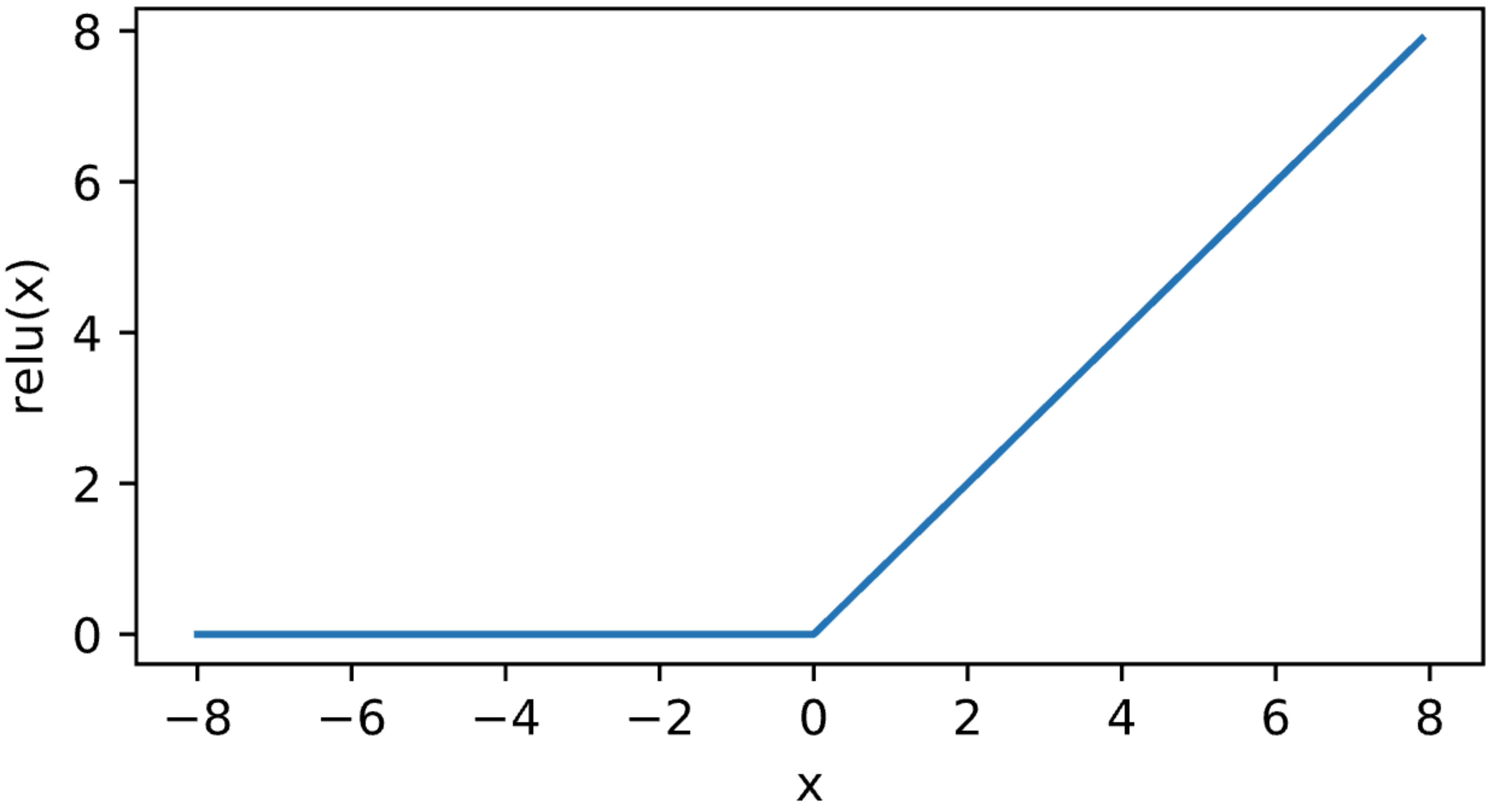
$$tanh(z) = 2sigmoid(2z) - 1$$



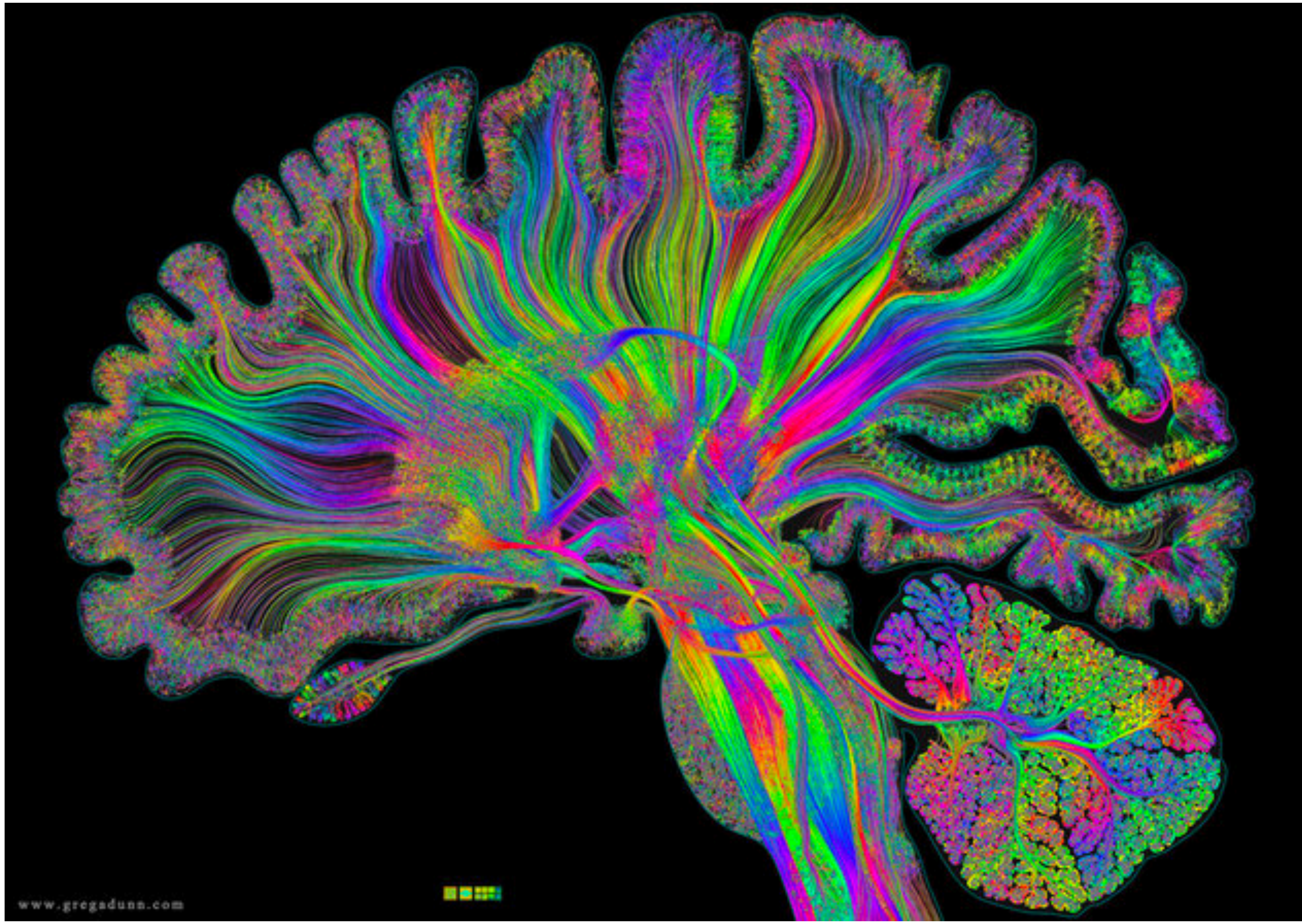
ReLU Activation

ReLU: rectified linear unit (commonly used in modern neural networks)

$$\mathrm{ReLU}(x) = \max(x, 0)$$



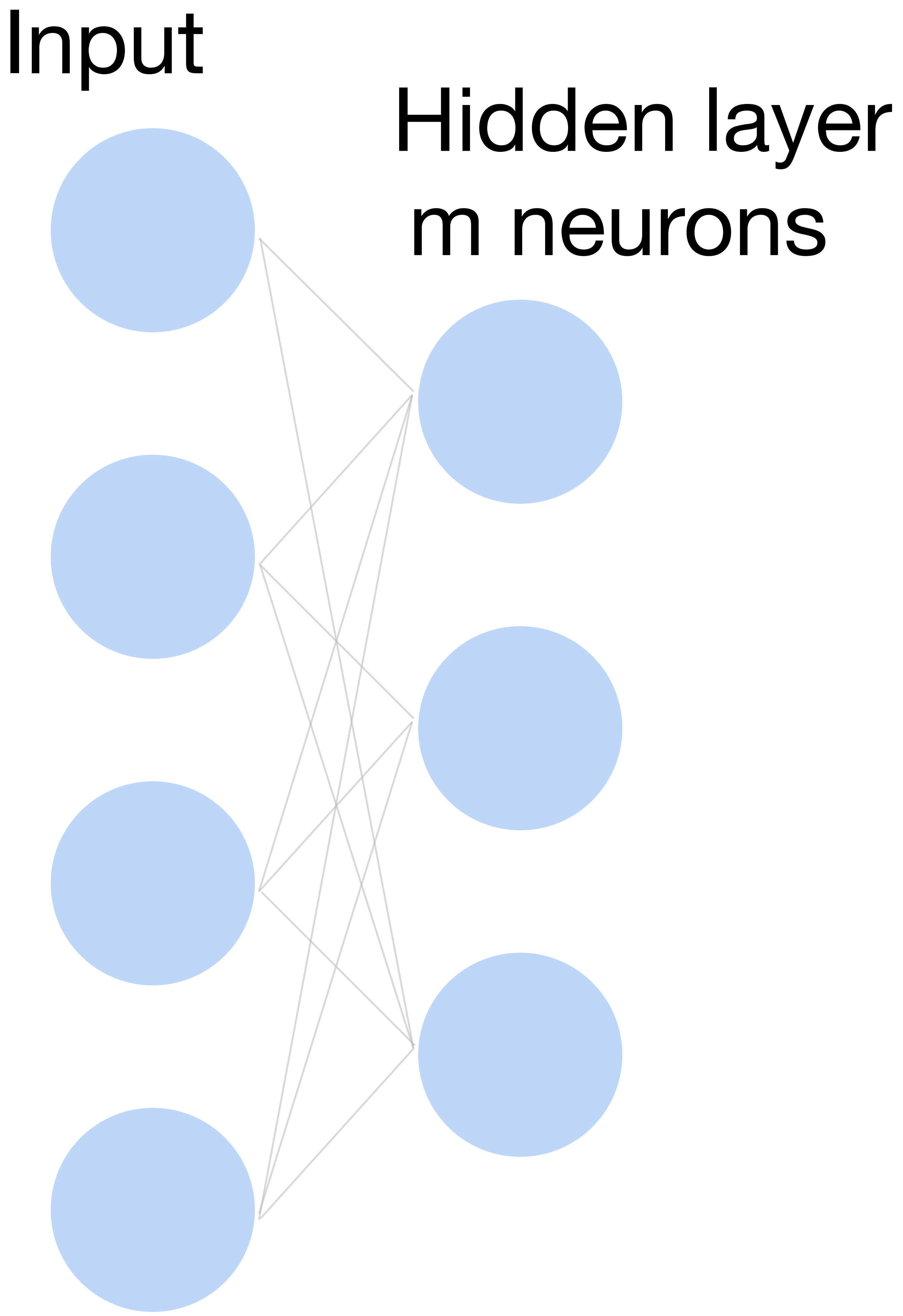
Multilayer Perceptron



Single Hidden Layer

How to classify

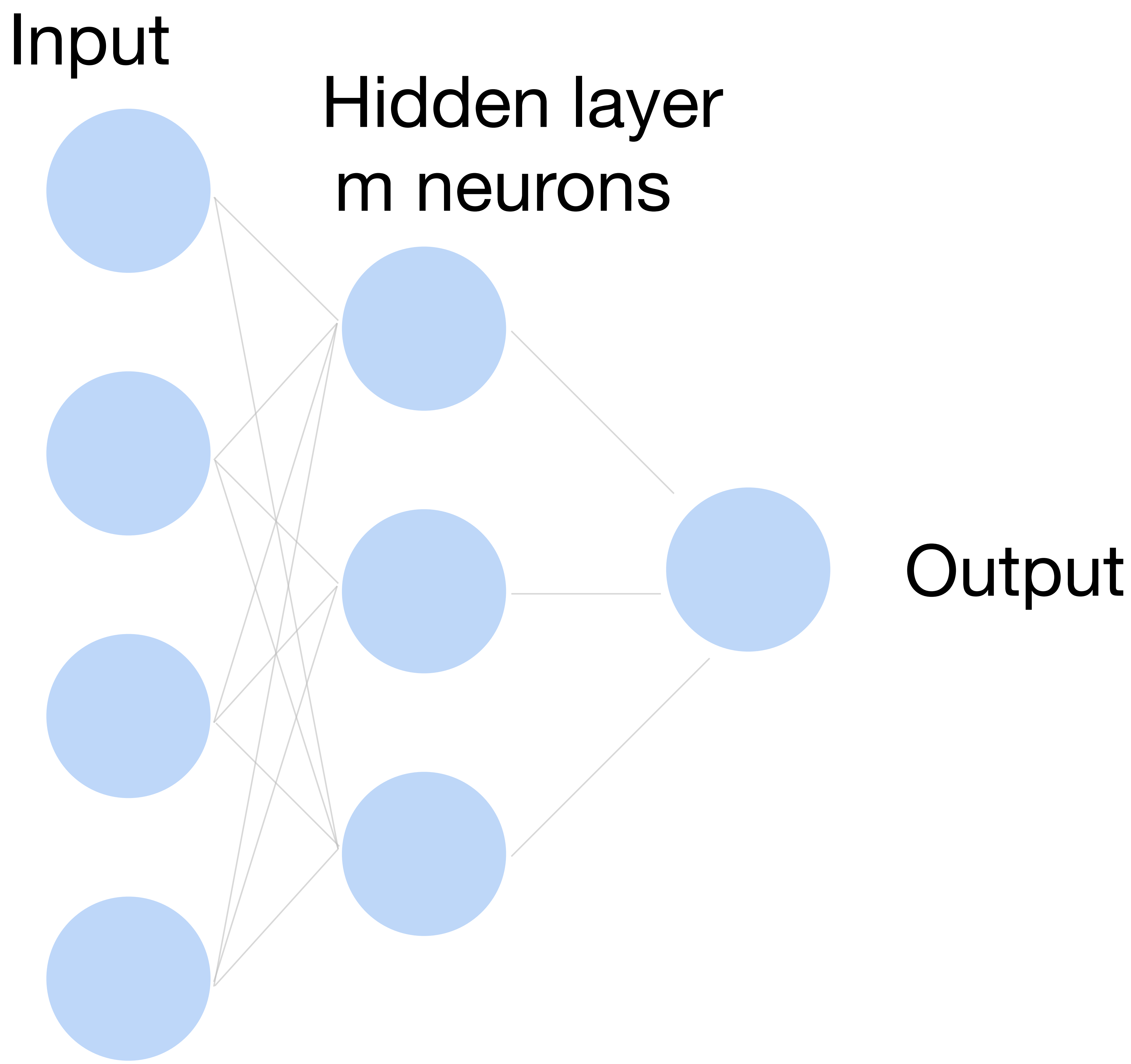
Cats vs. dogs?



Single Hidden Layer

How to classify

Cats vs. dogs?

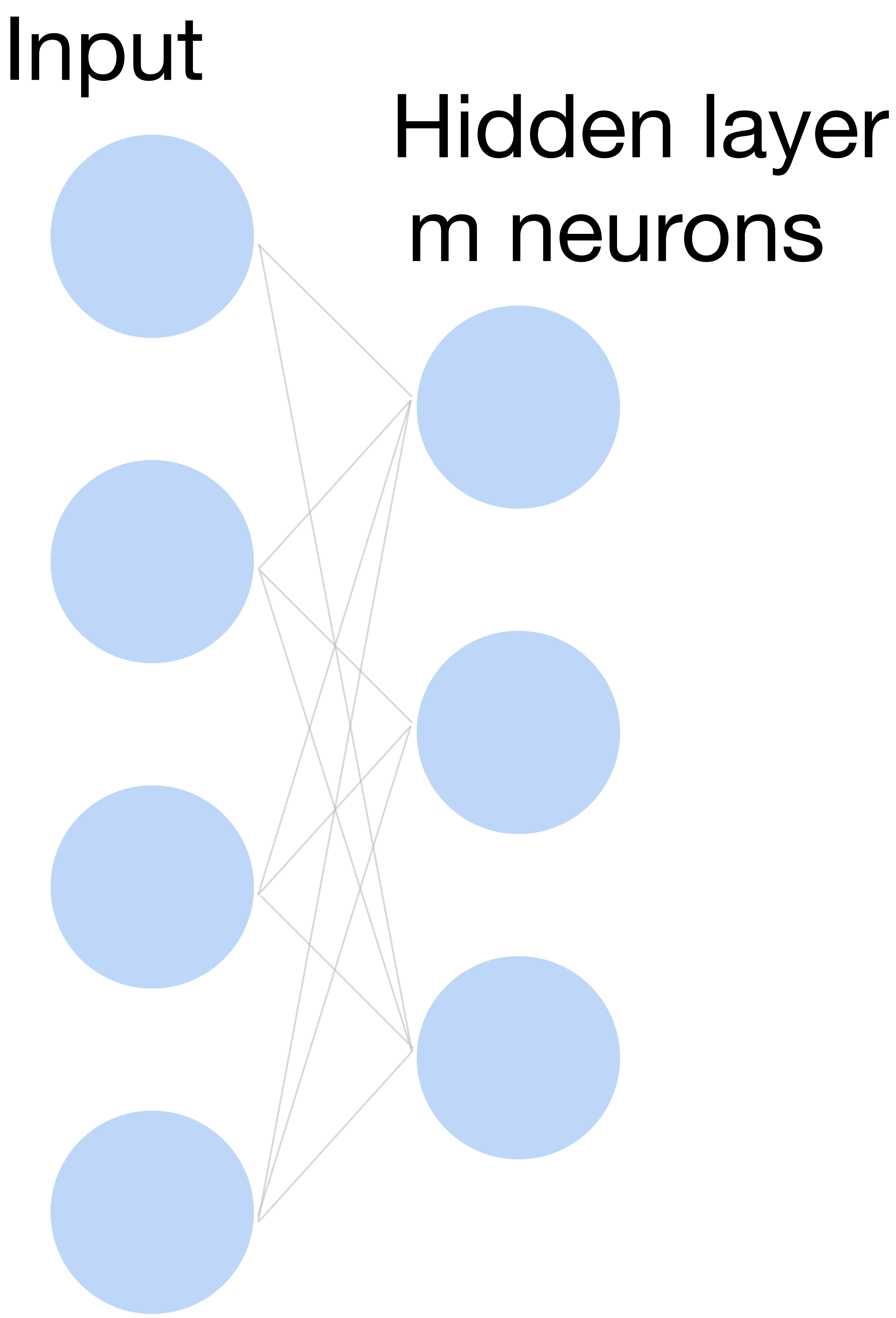


Single Hidden Layer

- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

σ is an element-wise activation function

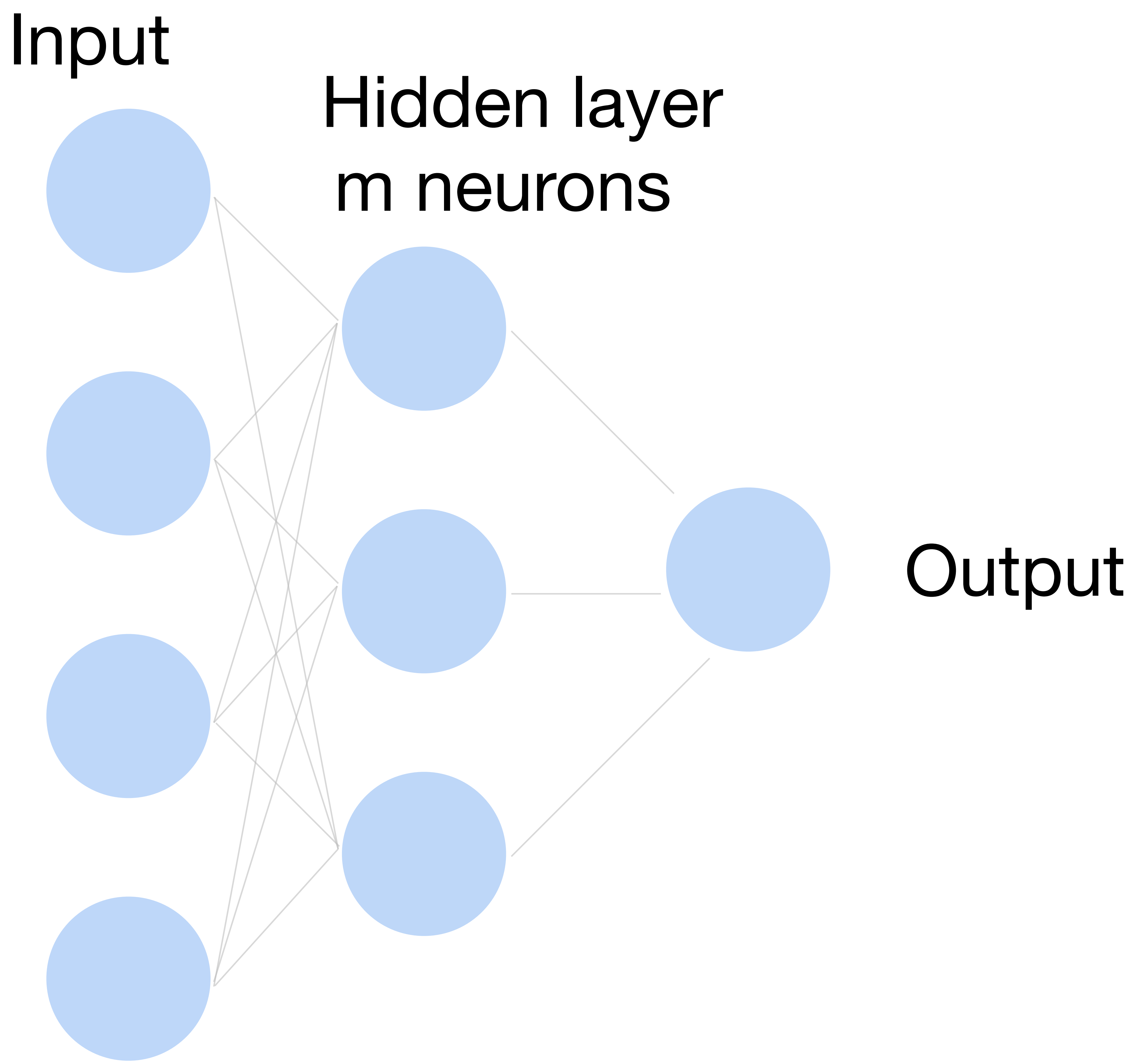


Input

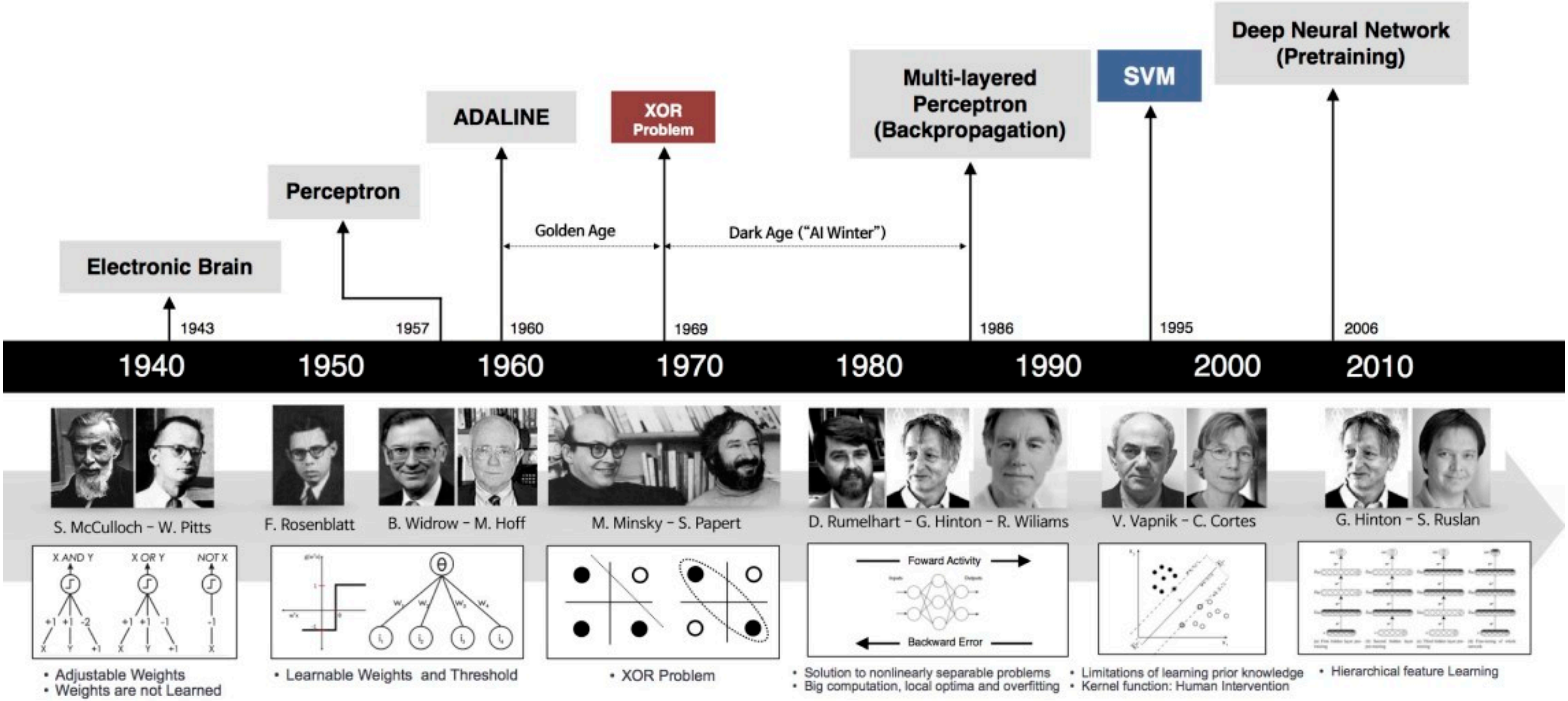
Hidden layer
m neurons

Single Hidden Layer

- Output $\mathbf{f} = \mathbf{w}_2^\top \mathbf{h} + b_2$



Brief history of neural networks



Electronic Brain 1943

Perceptron 1957

ADALINE 1960

XOR Problem 1969

Multi-layered Perceptron (Backpropagation) 1986

SVM 1995

Deep Neural Network (Pretraining) 2006

Golden Age

Dark Age ("AI Winter")

1940 1950 1960 1970 1980 1990 2000 2010

S. McCulloch - W. Pitts

- Adjustable Weights
- Weights are not Learned

F. Rosenblatt

B. Widrow - M. Hoff

- Learnable Weights and Threshold

M. Minsky - S. Papert

- XOR Problem

D. Rumelhart - G. Hinton - R. Wiliams

- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting

V. Vapnik - C. Cortes

- Limitations of learning prior knowledge
- Kernel function: Human Intervention

G. Hinton - S. Ruslan

- Hierarchical feature Learning

What we’ve learned today…

- Single-layer Perceptron
    - Motivation
    - Activation function
    - Representing AND, OR, NOT
- Brief history of neural networks